MQL5 언어 레퍼런스

MetaTrader 5 클라이언트 터미널에서 사용 가능합니다

MQL5를 학습해 보세요
그리고 어떠한 문제라도 해결해 보세요:

- 어떠한 난이도라도 가능한
  나만의 기술적 지표를 생성해 보세요
- 자동 트레이딩을 사용해 보세요 – 다양한 금융
  시장에 대응하는
  트레이딩 시스템을 자동화 하세요
- 수학적 연구나 전통적 방법에 기초한
  나만의 분석 툴을 만들어 보세요
- 다양한 영역(트레이딩, 모니터링, 알리미, 등)의
  문제를 해결하는 유용한 트레이딩 시스템을
  만들어 보세요

© 2000-2025, MetaQuotes Ltd

Content

MQL5 リファレンス 73

1 言語基礎 76

構文 77

コメント 78

識別子 79

予約語 80

データ型 82

整数型 83

Char、Short、Int 及び Long 型 85

文字定数 89

日付時刻型 92

カラー型 93

ブール型 94

列挙型 95

浮動小数点数型（double、float） 97

복소수(complex) 103

文字列型 104

構造体、クラス、インターフェイス 105

動的配列オブジェクト 131

行列とベクトル 132

型キャスト 138

Void 型とNULL 定数 143

ユーザー定義型 144

オブジェクトポインタ 154

参照： 修飾子「 & 」とキーワード「 this 」 158

演算子と式 160

式 161

算術演算子 162

代入演算子 163

関係の演算子 164

ブール演算子 165

ビット演算子 167

その他の演算子 170

優先順位のルール 174

演算子 176

重文演算子 177

式の演算子 178

Return 演算子 179

if-else 条件演算子 180

三項演算子 ?: 181

Switch 演算子 183

while 反復演算子 185

for 反復演算子 186

do while 反復演算子 187

Break 演算子 188

Continue 演算子 189

행렬 곱 연산자 @ 190

new オブジェクト作成演算子 192

delete オブジェクト解除演算子 193

함수 194

関数の呼び出し 196

パラメータの引き渡し 197

関数の多重定義（オーバーロード） 200

© 2000-2025, MetaQuotes Ltd.

演算子の多重定義（オーバーロード） 203
外部関数の記述 216
関数のエクスポート 218
イベント処理の関数 219
변수 230
ローカル変数 234
仮引数 236
静的変数 238
グローバル変数 239
Input変数 240
Extern 変数 246
変数の初期化 247
変数のアクセス権の有効範囲とライフタイム 249
オブジェクトの作成と解除 251
プリプロセッサ 253
マクロ代入（ #define ） 255
プログラムのプロパティ（#property） 258
ファイルの読み込み（#include） 263
関数のインポート（ #import ） 264
条件付きコンパイル（ #ifdef、#ifndef、#else、#endif ） 267
オブジェクト指向プログラミング 269
型の封装と拡張 271
継承 273
ポリモーフィズム 278
多重定義（オーバーロード） 282
仮想関数 283
クラスの静的メンバ 287
関数テンプレート 291
クラステンプレート 295
抽象クラス 300
名前空間 302
2 標準的な定数、列挙と構造体 305
チャート定数 306
チャートイベントの種類 307
チャート時間軸 314
チャートプロパティ 316
位置定数 323
チャート表示 324
チャート操作の例 326
オブジェクト定数 384
オブジェクト型 385
OBJ_VLINE 387
OBJ_HLINE 392
OBJ_TREND 397
OBJ_TRENDBYANGLE 404
OBJ_CYCLES 410
OBJ_ARROWED_LINE 416
OBJ_CHANNEL 422
OBJ_STDDEVCHANNEL 429
OBJ_REGRESSION 435
OBJ_PITCHFORK 441
OBJ_GANNLINE 449
OBJ_GANNFAN 456
OBJ_GANNGRID 463
OBJ_FIBO 470
OBJ_FIBOTIMES 477
OBJ_FIBOFAN 484
OBJ_FIBOARC 491

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOCHANNEL ... 498
OBJ_EXPANSION ... 506
OBJ_ELLIOTWAVE5 ... 514
OBJ_ELLIOTWAVE3 ... 522
OBJ_RECTANGLE ... 529
OBJ_TRIANGLE ... 535
OBJ_ELLIPSE ... 542
OBJ_ARROW_THUMB_UP ... 548
OBJ_ARROW_THUMB_DOWN ... 554
OBJ_ARROW_UP ... 560
OBJ_ARROW_DOWN ... 566
OBJ_ARROW_STOP ... 572
OBJ_ARROW_CHECK ... 578
OBJ_ARROW_LEFT_PRICE ... 584
OBJ_ARROW_RIGHT_PRICE ... 589
OBJ_ARROW_BUY ... 594
OBJ_ARROW_SELL ... 599
OBJ_ARROW ... 604
OBJ_TEXT ... 610
OBJ_LABEL ... 616
OBJ_BUTTON ... 624
OBJ_CHART ... 631
OBJ_BITMAP ... 638
OBJ_BITMAP_LABEL ... 645
OBJ_EDIT ... 652
OBJ_EVENT ... 659
OBJ_RECTANGLE_LABEL ... 664
オブジェクトプロパティ ... 670
オブジェクト結合の方法 ... 690
チャートのコーナー ... 695
オブジェクトの可視性 ... 698
エリオット波動のレベル ... 701
ギャンオブジェクト ... 702
ウェブカラー ... 704
特殊文字 ... 706
指標定数 ... 707
価格定数 ... 708
平滑化の方法 ... 711
指標線 ... 712
描画スタイル ... 714
カスタム指標プロパティ ... 719
指標の種類 ... 723
データ型識別子 ... 725
環境状態 ... 726
クライアント端末プロパティ ... 727
MQL5 プログラム**実行**プロパティ ... 736
シンボルプロパティ ... 740
口座プロパティ ... 795
テスト統計 ... 802
取引定数 ... 806
履歴データベースプロパティ ... 807
注文プロパティ ... 808
ポジションプロパティ ... 813
約定プロパティ ... 817
取引操作の種類 ... 821
取引トランザクションの種類 ... 833
DOM での取引注文 ... 835
シグナルプロパティ ... 836

© 2000-2025, MetaQuotes Ltd.

名前付き定数 838
定義済みマクロ代入 839
数学定数 845
数値定数 846
初期化解除の理由のコード 849
オブジェクトポインタのチェック 851
その他の定数 852
データ構造体 856
データ型構造体 857
入力パラメータの構造体 858
履歴データの構造体 859
DOM構造体 860
取引リクエスト構造体 861
小切手リクエスト構造体 874
取引リクエスト結果の構造体 875
取引処理構造体 878
現在価格の構造体 885
経済指標カレンダー構造体 886
エラーコードと警告コード 895
取引サーバのリターンコード 896
コンパイラ警告 899
は、コンパイルエラー 902
ランタイムエラー 912
入出力定数 925
ファイルを開く際のフラグ 926
ファイルプロパティ 928
ファイル内の位置 929
コードページの利用 930
メッセージボックス 931
3 MQL5 プログラム 933
プログラム実行 934
取引許可 941
クライアント端末イベント 945
リソース 948
インポートした関数の呼び出し 959
ランタイムエラー 961
取引ストラテジーのテスト 962
4 定義済み変数 986
_AppliedTo 987
_Digits 989
_Point 990
_LastError 991
_Period 992
_RandomSeed 993
_StopFlag 994
_Symbol 995
_UninitReason 996
_IsX64 997
5 共通関数 998
Alert 1000
CheckPointer 1005
Comment 1007
CryptEncode 1008
CryptDecode 1010
DebugBreak 1012
ExpertRemove 1013
GetPointer 1015

© 2000-2025, MetaQuotes Ltd.

GetTickCount 1019
GetTickCount64 1020
GetMicrosecondCount 1022
MessageBox 1024
PeriodSeconds 1029
PlaySound 1031
Print 1034
PrintFormat 1036
ResetLastError 1041
ResourceCreate 1042
ResourceFree 1046
ResourceReadImage 1049
ResourceSave 1053
SetReturnError 1058
SetUserError 1060
Sleep 1061
TerminalClose 1063
TesterHideIndicators 1065
TesterStatistics 1067
TesterStop 1069
TesterDeposit 1071
TesterWithdrawal 1074
TranslateKey 1077
ZeroMemory 1078
6 配列関数 1081
ArrayBsearch 1083
ArrayCopy 1087
ArrayCompare 1092
ArrayFree 1095
ArrayGetAsSeries 1104
ArrayInitialize 1107
ArrayFill 1109
ArrayIsDynamic 1111
ArrayIsSeries 1113
ArrayMaximum 1115
ArrayMinimum 1126
ArrayPrint 1137
ArrayRange 1139
ArrayResize 1140
ArrayInsert 1143
ArrayRemove 1145
ArrayReverse 1147
ArraySetAsSeries 1149
ArraySize 1152
ArraySort 1154
ArraySwap 1159
ArrayToFP16 1161
ArrayToFP8 1164
ArrayFromFP16 1167
ArrayFromFP8 1170
7 행렬과 벡터 메서드 1173
행렬 및 벡터 유형 1179
열거형 1180
초기화 1185
Assign 1188
CopyIndicatorBuffer 1190
CopyRates 1192
CopyTicks 1197

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange 1200
Eye 1202
Identity 1204
Ones 1206
Zeros 1207
Full 1208
Tri 1209
Init 1210
Fill 1212
Random 1213
행렬 및 벡터 조작 1215
HasNan 1216
ReplaceNan 1217
Transpose 1218
TransposeConjugate 1220
TriL 1221
TriU 1222
Diag 1223
Row 1225
Col 1227
Copy 1229
Conjugate 1231
Concat 1232
Compare 1234
CompareByDigits 1236
CompareEqual 1238
Flat 1241
Clip 1243
Reshape 1244
Resize 1246
Set 1248
SwapRows 1250
SwapCols 1251
Split 1252
Hsplit 1254
Vsplit 1256
ArgSort 1258
Sort 1259
연산 1261
수학 연산 1263
함수 1264
행렬 및 벡터 곱 1265
MatMul 1266
GeMM 1271
Power 1275
Dot 1278
Kron 1280
Inner 1282
Outer 1284
CorrCoef 1287
Cov 1290
Correlate 1293
Convolve 1296
변환 1299
Cholesky 1300
Eig 1301
EigVals 1304
LU 1305

© 2000-2025, MetaQuotes Ltd.

LUP 1307
QR 1309
SVD 1311
통계 1314
ArgMax 1315
ArgMin 1316
Max 1317
Min 1318
Ptp 1319
Sum 1320
Prod 1321
CumSum 1323
CumProd 1325
Percentile 1327
Quantile 1329
Median 1331
Mean 1333
Average 1334
Std 1336
Var 1338
LinearRegression 1340
기능 1343
Rows 1344
Cols 1345
Size 1346
Norm 1347
Cond 1350
Det 1353
SLogDet 1355
Rank 1356
Trace 1359
Spectrum 1360
Matrix Classification 1361
IsSymmetric 1362
IsHermitian 1363
IsUpperTriangular 1364
IsLowerTriangular 1365
IsTrapezoidal 1366
IsUpperHessenberg 1367
IsLowerHessenberg 1368
IsTridiagonal 1369
IsUpperBidiagonal 1370
IsLowerBidiagonal 1371
IsDiagonal 1372
IsScalar 1373
솔루션 1374
Solve 1375
LstSq 1376
Inv 1377
PInv 1379
머신 러닝 1380
Activation 1385
Derivative 1389
Loss 1391
LossGradient 1393
RegressionMetric 1395
ConfusionMatrix 1397
ConfusionMatrixMultilabel 1399

© 2000-2025, MetaQuotes Ltd.

ClassificationMetric 1401
ClassificationScore 1405
PrecisionRecall 1409
ReceiverOperatingCharacteristic 1413
OpenBLAS 1417
Singular Value Decomposition 1420
SingularValueDecompositionDC 1421
SingularValueDecompositionQR 1423
SingularValueDecompositionQRPivot 1425
SingularValueDecompositionBisect 1429
SingularValueDecompositionJacobiHigh 1432
SingularValueDecompositionJacobiLow 1437
SingularValueDecompositionBidiagDC 1440
SingularValueDecompositionBidiagBisect 1443
SingularValueDecompositionBidiagQR 1446
Eigen Values 1448
General Matrices 1452
EigenSolver 1454
EigenSolverX 1456
EigenSolverSchur 1460
EigenSolver2 1462
EigenSolver2X 1465
EigenSolver2Schur 1470
EigenSolver2Blocked 1473
EigenSolver2SchurBlocked 1476
EigenHessenbergSchurQ 1479
EigenVectorsTriangularZ 1481
EigenVectorsTriangularZBlocked 1483
Symmetric Matrices 1485
EigenSymmetricDC 1487
EigenSymmetricQR 1489
EigenSymmetricRobust 1491
EigenSymmetricBisect 1494
EigenSymmetricDC2s 1497
EigenSymmetricQR2s 1499
EigenSymmetricRobust2s 1501
EigenSymmetricBisect2s 1504
EigenSymmetric2DC 1507
EigenSymmetric2QR 1510
EigenSymmetric2Bisect 1513
Tridiagonal Matrices 1517
EigenTridiagonalDC 1519
EigenTridiagonalQR 1521
EigenTridiagonalRobust 1523
EigenTridiagonalBisect 1526
EigenTridiagonalQL 1529
EigenTridiagonalDCQ 1530
EigenTridiagonalQRQ 1532
EigenTridiagonalPosDefQ 1534
Linear Equations 1536
LinearEquationsSolution 1538
LinearEquationsSolutionTriangular 1540
InverseTriangular 1542
CondNumReciprocalTriangular 1543
LinearEquationsSolutionSy 1545
LinearEquationsSolutionComplexSy 1547
LinearEquationsSolutionSyPD 1548
LinearEquationsSolutionGeTrid 1550

© 2000-2025, MetaQuotes Ltd.

LinearEquationsSolutionSyTridPD ... 1552
SylvesterEquationTriangular ... 1554
SylvesterEquationTriangularBlocked ... 1557
Least Squares ... 1559
LeastSquaresSolution ... 1561
LeastSquaresSolutionDC ... 1564
LeastSquaresSolutionSVD ... 1567
LeastSquaresSolutionWY ... 1570
LeastSquaresSolutionsQRPivot ... 1573
LeastSquaresSolutionQRTallSkinny ... 1576
Orthogonal Factorizations ... 1579
FactorizationQR ... 1580
FactorizationQRNonNeg ... 1582
FactorizationQRPivot ... 1584
FactorizationLQ ... 1586
FactorizationQL ... 1588
FactorizationRQ ... 1590
Factorizations ... 1592
FactorizationPLU ... 1594
FactorizationPLUQ ... 1596
FactorizationPLUGeTrid ... 1598
FactorizationLDL ... 1600
FactorizationLDLComplexSy ... 1602
FactorizationLDLSyTridPD ... 1603
FactorizationCholesky ... 1605
FactorizationCholeskySyPS ... 1606
FactorizationPLURaw ... 1608
FactorizationPLUQRaw ... 1610
FactorizationPLUGeTridRaw ... 1612
FactorizationLDLRaw ... 1614
Factored Calculations ... 1616
PLULinearEquationsSolution ... 1620
PLUInverse ... 1623
PLUCondNumReciprocal ... 1624
PLUQLinearEquationsSolution ... 1626
PLUGeTridLinearEquationsSolution ... 1629
PLUGeTridCondNumReciprocal ... 1632
LDLLinearEquationsSolution ... 1634
LDLInverse ... 1636
LDLCondNumReciprocal ... 1637
LDLSyTridPDLinearEquationsSolution ... 1639
LDLSyTridPDCondNumReciprocal ... 1641
CholeskyLinearEquationsSolution ... 1643
CholeskyInverse ... 1645
CholeskyCondNumReciprocal ... 1646
SylvesterEquationSchur ... 1648
SylvesterEquationSchurBlocked ... 1650
Pseudo Inverse ... 1652
Polar Decomposition ... 1654
Dynamic Mode Decomposition ... 1656
DynamicModeDecomposition ... 1657
DynamicModeDecompositionQR ... 1663
Singular Spectrum Analysis ... 1670
SingularSpectrumAnalysisSpectrum ... 1671
SingularSpectrumAnalysisForecast ... 1672
SingularSpectrumAnalysisReconstructComponents ... 1673
SingularSpectrumAnalysisReconstructSeries ... 1674
Reductions ... 1675

© 2000-2025, MetaQuotes Ltd.

ReduceToBidiagonal 1676
ReflectBidiagonalToQP 1679
ReduceSymmetricToTridiagonal 1681
ReflectTridiagonalToQ 1683
ReduceToHessenberg 1685
ReflectHessenbergToQ 1687
Matrix Norm 1688
MatrixNorm 1690
MatrixNormGeTrid 1692
MatrixNormHessenberg 1694
MatrixNormSy 1696
MatrixNormComplexSy 1698
MatrixNormSyTrid 1699
MatrixNormTriangular 1701
Matrix Balance 1703
MatrixBalance 1705
EigenVectorsBackward 1707
ReduceToHessenbergBalanced 1710
ReflectHessenbergBalancedToQ 1712
EigenHessenbergBalancedSchurQ 1714
8 変換関数 1716
CharToString 1717
CharArrayToString 1719
CharArrayToStruct 1721
StructToCharArray 1725
ColorToARGB 1728
ColorToString 1730
DoubleToString 1731
EnumToString 1732
IntegerToString 1734
ShortToString 1736
ShortArrayToString 1738
TimeToString 1740
NormalizeDouble 1743
StringToCharArray 1745
StringToColor 1747
StringToDouble 1748
StringToInteger 1749
StringToShortArray 1750
StringToTime 1752
StringFormat 1755
9 산술 함수 1759
MathAbs 1761
MathArccos 1762
MathArcsin 1765
MathArctan 1768
MathArctan2 1771
MathClassify 1775
MathCeil 1777
MathCos 1779
MathExp 1782
MathFloor 1785
MathLog 1787
MathLog10 1791
MathMax 1795
MathMin 1797
MathMod 1799
MathPow 1800

© 2000-2025, MetaQuotes Ltd.

MathRand 1804
MathRound 1805
MathSin 1807
MathSqrt 1810
MathSrand 1814
MathTan 1817
MathIsValidNumber 1820
MathExpm1 1821
MathLog1p 1824
MathArccosh 1828
MathArcsinh 1831
MathArctanh 1834
MathCosh 1837
MathSinh 1840
MathTanh 1843
MathSwap 1846
10 文字列関数 1849
StringAdd 1850
StringBufferLen 1852
StringCompare 1853
StringConcatenate 1855
StringFill 1857
StringFind 1858
StringGetCharacter 1861
StringInit 1863
StringLen 1864
StringSetLength 1865
StringReplace 1866
StringReserve 1867
StringSetCharacter 1869
StringSplit 1871
StringSubstr 1872
StringToLower 1873
StringToUpper 1874
StringTrimLeft 1875
StringTrimRight 1876
11 日付と時刻 1877
TimeCurrent 1878
TimeTradeServer 1880
TimeLocal 1882
TimeGMT 1884
TimeDaylightSavings 1886
TimeGMTOffset 1887
TimeToStruct 1888
StructToTime 1889
12 口座情報 1891
AccountInfoDouble 1892
AccountInfoInteger 1893
AccountInfoString 1895
13 チェックアップ 1896
GetLastError 1897
IsStopped 1898
UninitializeReason 1899
TerminalInfoInteger 1901
TerminalInfoDouble 1902
TerminalInfoString 1903
MQLInfoInteger 1904

© 2000-2025, MetaQuotes Ltd.

MQLInfoString 1905
Symbol 1906
Period 1907
Digits 1908
Point 1909
14 イベント処理 1910
OnStart 1912
OnInit 1915
OnDeinit 1918
OnTick 1921
OnCalculate 1927
OnTimer 1931
OnTrade 1934
OnTradeTransaction 1939
OnBookEvent 1945
OnChartEvent 1948
OnTester 1954
OnTesterInit 1961
OnTesterDeinit 1968
OnTesterPass 1969
15 市場情報 1970
SymbolsTotal 1971
SymbolExist 1972
SymbolName 1974
SymbolSelect 1976
SymbolIsSynchronized 1978
SymbolInfoDouble 1979
SymbolInfoInteger 1981
SymbolInfoString 1983
SymbolInfoMarginRate 1985
SymbolInfoTick 1987
SymbolInfoSessionQuote 1988
SymbolInfoSessionTrade 1990
MarketBookAdd 1992
MarketBookRelease 1994
MarketBookGet 1996
16 経済指標カレンダー 1997
CalendarCountryById 1998
CalendarEventById 2000
CalendarValueById 2003
CalendarCountries 2006
CalendarEventByCountry 2008
CalendarEventByCurrency 2010
CalendarValueHistoryByEvent 2012
CalendarValueHistory 2015
CalendarValueLastByEvent 2018
CalendarValueLast 2023
17 時系列と指標へのアクセス 2027
配列、バッファ及び時系列における索引付けの方向 2031
データアクセスの整理 2034
SeriesInfoInteger 2043
Bars 2045
BarsCalculated 2048
IndicatorCreate 2050
IndicatorParameters 2052
IndicatorRelease 2054
CopyBuffer 2056

© 2000-2025, MetaQuotes Ltd.

CopyRates 2060
CopySeries 2064
CopyTime 2068
CopyOpen 2071
CopyHigh 2074
CopyLow 2078
CopyClose 2081
CopyTickVolume 2084
CopyRealVolume 2088
CopySpread 2091
CopyTicks 2095
CopyTicksRange 2101
iBars 2103
iBarShift 2104
iClose 2107
iHigh 2109
iHighest 2111
iLow 2112
iLowest 2114
iOpen 2115
iTime 2117
iTickVolume 2119
iRealVolume 2121
iVolume 2123
iSpread 2125
18 カスタム銘柄 2127
CustomSymbolCreate 2128
CustomSymbolDelete 2131
CustomSymbolSetInteger 2133
CustomSymbolSetDouble 2137
CustomSymbolSetString 2141
CustomSymbolSetMarginRate 2145
CustomSymbolSetSessionQuote 2149
CustomSymbolSetSessionTrade 2154
CustomRatesDelete 2159
CustomRatesReplace 2164
CustomRatesUpdate 2169
CustomTicksAdd 2174
CustomTicksDelete 2182
CustomTicksReplace 2190
CustomBookAdd 2199
19 チャート操作 2202
ChartApplyTemplate 2204
ChartSaveTemplate 2207
ChartWindowFind 2212
ChartTimePriceToXY 2214
ChartXYToTimePrice 2216
ChartOpen 2218
ChartFirst 2220
ChartNext 2221
ChartClose 2222
ChartSymbol 2224
ChartPeriod 2225
ChartRedraw 2226
ChartSetDouble 2229
ChartSetInteger 2231
ChartSetString 2233
ChartGetDouble 2235

© 2000-2025, MetaQuotes Ltd.

ChartGetInteger 2237
ChartGetString 2239
ChartNavigate 2241
ChartID 2244
ChartIndicatorAdd 2246
ChartIndicatorDelete 2250
ChartIndicatorGet 2253
ChartIndicatorName 2255
ChartIndicatorsTotal 2257
ChartWindowOnDropped 2259
ChartPriceOnDropped 2260
ChartTimeOnDropped 2261
ChartXOnDropped 2262
ChartYOnDropped 2263
ChartSetSymbolPeriod 2264
ChartScreenShot 2266
20 取引関数 2269
OrderCalcMargin 2271
OrderCalcProfit 2275
OrderCheck 2278
OrderSend 2282
OrderSendAsync 2287
PositionsTotal 2298
PositionGetSymbol 2299
PositionSelect 2301
PositionSelectByTicket 2303
PositionGetDouble 2305
PositionGetInteger 2308
PositionGetString 2310
PositionGetTicket 2313
OrdersTotal 2315
OrderGetTicket 2316
OrderSelect 2318
OrderGetDouble 2322
OrderGetInteger 2326
OrderGetString 2332
HistorySelect 2335
HistorySelectByPosition 2337
HistoryOrderSelect 2341
HistoryOrdersTotal 2344
HistoryOrderGetTicket 2345
HistoryOrderGetDouble 2347
HistoryOrderGetInteger 2351
HistoryOrderGetString 2354
HistoryDealSelect 2357
HistoryDealsTotal 2360
HistoryDealGetTicket 2361
HistoryDealGetDouble 2363
HistoryDealGetInteger 2368
HistoryDealGetString 2371
21 売買シグナル 2375
SignalBaseGetDouble 2376
SignalBaseGetInteger 2377
SignalBaseGetString 2378
SignalBaseSelect 2379
SignalBaseTotal 2380
SignalInfoGetDouble 2381
SignalInfoGetInteger 2382

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetString 2383
SignalInfoSetDouble 2384
SignalInfoSetInteger 2385
SignalSubscribe 2386
SignalUnSubscribe 2387
22 ネットワーク関数 2388
SocketCreate 2390
SocketClose 2393
SocketConnect 2396
SocketIsConnected 2400
SocketIsReadable 2404
SocketIsWritable 2407
SocketTimeouts 2411
SocketRead 2413
SocketSend 2417
SocketTlsHandshake 2421
SocketTlsCertificate 2426
SocketTlsRead 2430
SocketTlsReadAvailable 2434
SocketTlsSend 2439
WebRequest 2443
SendFTP 2446
SendMail 2448
SendNotification 2450
23 端末のグローバル変数 2452
GlobalVariableCheck 2453
GlobalVariableTime 2454
GlobalVariableDel 2455
GlobalVariableGet 2456
GlobalVariableName 2457
GlobalVariableSet 2458
GlobalVariablesFlush 2459
GlobalVariableTemp 2460
GlobalVariableSetOnCondition 2461
GlobalVariablesDeleteAll 2462
GlobalVariablesTotal 2463
24 ファイル関数 2464
FileSelectDialog 2467
FileFindFirst 2469
FileFindNext 2471
FileFindClose 2473
FileIsExist 2475
FileOpen 2478
FileClose 2481
FileCopy 2482
FileDelete 2485
FileMove 2487
FileFlush 2489
FileGetInteger 2491
FileIsEnding 2494
FileIsLineEnding 2495
FileReadArray 2500
FileReadBool 2502
FileReadDatetime 2505
FileReadDouble 2508
FileReadFloat 2511
FileReadInteger 2514

© 2000-2025, MetaQuotes Ltd.

FileReadLong 2518
FileReadNumber 2521
FileReadString 2526
FileReadStruct 2528
FileSeek 2532
FileSize 2535
FileTell 2537
FileWrite 2540
FileWriteArray 2543
FileWriteDouble 2546
FileWriteFloat 2549
FileWriteInteger 2551
FileWriteLong 2554
FileWriteString 2556
FileWriteStruct 2559
FileLoad 2562
FileSave 2564
FolderCreate 2566
FolderDelete 2569
FolderClean 2572
25 カスタム指標 2575
指標スタイルの例 2578
DRAW_NONE 2587
DRAW_LINE 2590
DRAW_SECTION 2594
DRAW_HISTOGRAM 2598
DRAW_HISTOGRAM2 2602
DRAW_ARROW 2606
DRAW_ZIGZAG 2611
DRAW_FILLING 2616
DRAW_BARS 2621
DRAW_CANDLES 2627
DRAW_COLOR_LINE 2633
DRAW_COLOR_SECTION 2638
DRAW_COLOR_HISTOGRAM 2643
DRAW_COLOR_HISTOGRAM2 2648
DRAW_COLOR_ARROW 2653
DRAW_COLOR_ZIGZAG 2659
DRAW_COLOR_BARS 2664
DRAW_COLOR_CANDLES 2670
指標プロパティと関数との関係 2676
SetIndexBuffer 2678
IndicatorSetDouble 2681
IndicatorSetInteger 2685
IndicatorSetString 2689
PlotIndexSetDouble 2692
PlotIndexSetInteger 2693
PlotIndexSetString 2697
PlotIndexGetInteger 2698
26 オブジェクト関数 2701
ObjectCreate 2703
ObjectName 2707
ObjectDelete 2708
ObjectsDeleteAll 2709
ObjectFind 2710
ObjectGetTimeByValue 2711
ObjectGetValueByTime 2712
ObjectMove 2713

© 2000-2025, MetaQuotes Ltd.

ObjectsTotal 2714
ObjectSetDouble 2715
ObjectSetInteger 2718
ObjectSetString 2721
ObjectGetDouble 2723
ObjectGetInteger 2725
ObjectGetString 2727
TextSetFont 2729
TextOut 2731
TextGetSize 2735
27 テクニカル指標 2736
iAC 2739
iAD 2744
iADX 2749
iADXWilder 2754
iAlligator 2759
iAMA 2766
iAO 2771
iATR 2776
iBearsPower 2781
iBands 2786
iBullsPower 2792
iCCI 2797
iChaikin 2802
iCustom 2807
iDEMA 2810
iDeMarker 2815
iEnvelopes 2820
iForce 2826
iFractals 2831
iFrAMA 2836
iGator 2841
iIchimoku 2848
iBWMFI 2855
iMomentum 2860
iMFI 2865
iMA 2870
iOsMA 2875
iMACD 2880
iOBV 2886
iSAR 2891
iRSI 2896
iRVI 2901
iStdDev 2906
iStochastic 2911
iTEMA 2917
iTriX 2922
iWPR 2927
iVIDyA 2932
iVolumes 2937
28 最適化の結果の操作 2942
FrameFirst 2944
FrameFilter 2945
FrameNext 2946
FrameInputs 2947
FrameAdd 2948
ParameterGetRange 2949
ParameterSetRange 2952

© 2000-2025, MetaQuotes Ltd.

29 イベント操作 2954
EventSetMillisecondTimer 2955
EventSetTimer 2956
EventKillTimer 2957
EventChartCustom 2958
30 OpenCL 操作 2963
CLHandleType 2965
CLGetInfoInteger 2966
CLGetInfoString 2969
CLContextCreate 2972
CLContextFree 2973
CLGetDeviceInfo 2974
CLProgramCreate 2979
CLProgramFree 2984
CLKernelCreate 2985
CLKernelFree 2986
CLSetKernelArg 2987
CLSetKernelArgMem 2988
CLSetKernelArgMemLocal 2989
CLBufferCreate 2990
CLBufferFree 2991
CLBufferWrite 2992
CLBufferRead 2997
CLExecute 3001
CLExecutionStatus 3003
31 データベースの操作 3004
DatabaseOpen 3006
DatabaseClose 3008
DatabaseImport 3009
DatabaseExport 3012
DatabasePrint 3018
DatabaseTableExists 3023
DatabaseExecute 3024
DatabasePrepare 3036
DatabaseReset 3045
DatabaseBind 3051
DatabaseBindArray 3056
DatabaseRead 3061
DatabaseReadBind 3062
DatabaseFinalize 3066
DatabaseTransactionBegin 3067
DatabaseTransactionCommit 3072
DatabaseTransactionRollback 3073
DatabaseColumnsCount 3074
DatabaseColumnName 3075
DatabaseColumnType 3076
DatabaseColumnSize 3077
DatabaseColumnText 3078
DatabaseColumnInteger 3079
DatabaseColumnLong 3080
DatabaseColumnDouble 3081
DatabaseColumnBlob 3082
32 DirectXの操作 3083
DXContextCreate 3085
DXContextSetSize 3086
DXContextGetSize 3087
DXContextClearColors 3088

© 2000-2025, MetaQuotes Ltd.

DXContextClearDepth 3089
DXContextGetColors 3090
DXContextGetDepth 3091
DXBufferCreate 3092
DXTextureCreate 3093
DXInputCreate 3099
DXInputSet 3100
DXShaderCreate 3101
DXShaderSetLayout 3102
DXShaderInputsSet 3103
DXShaderTexturesSet 3104
DXDraw 3105
DXDrawIndexed 3106
DXPrimiveTopologySet 3107
DXBufferSet 3108
DXShaderSet 3109
DXHandleType 3110
DXRelease 3111
33 Python用MetaTrader 3112
initialize 3117
login 3119
shutdown 3122
version 3123
last_error 3125
account_info 3127
terminal_info 3130
symbols_total 3133
symbols_get 3134
symbol_info 3137
symbol_info_tick 3141
symbol_select 3143
market_book_add 3147
market_book_get 3148
market_book_release 3151
copy_rates_from 3152
copy_rates_from_pos 3156
copy_rates_range 3159
copy_ticks_from 3162
copy_ticks_range 3165
orders_total 3168
orders_get 3169
order_calc_margin 3172
order_calc_profit 3175
order_check 3178
order_send 3182
positions_total 3187
positions_get 3188
history_orders_total 3191
history_orders_get 3193
history_deals_total 3196
history_deals_get 3198
34 ONNX 모델 3202
ONNX 지원 3203
포맷 변환 3204
자동 데이터 유형 변환 3205
모델 생성 3208
모델 실행 3215
전략 테스터에서 실행 3221

© 2000-2025, MetaQuotes Ltd.

OnnxCreate 3226
OnnxCreateFromBuffer 3227
OnnxRelease 3228
OnnxRun 3229
OnnxGetInputCount 3232
OnnxGetOutputCount 3233
OnnxGetInputName 3234
OnnxGetOutputName 3235
OnnxGetInputTypeInfo 3236
OnnxGetOutputTypeInfo 3237
OnnxSetInputShape 3238
OnnxSetOutputShape 3239
데이터 구조 3240
35 標準ライブラリ 3244
数学 3245
統計 3246
統計的特性 3249
MathMean 3250
MathVariance 3251
MathSkewness 3252
MathKurtosis 3253
MathMoments 3254
MathMedian 3255
MathStandardDeviation 3256
MathAverageDeviation 3257
正規分布 3258
MathProbabilityDensityNormal 3262
MathCumulativeDistributionNormal 3264
MathQuantileNormal 3266
MathRandomNormal 3268
MathMomentsNormal 3269
対数正規分布 3270
MathProbabilityDensityLognormal 3274
MathCumulativeDistributionLognormal 3276
MathQuantileLognormal 3278
MathRandomLognormal 3280
MathMomentsLognormal 3281
ベータ分布 3282
MathProbabilityDensityBeta 3286
MathCumulativeDistributionBeta 3288
MathQuantileBeta 3290
MathRandomBeta 3292
MathMomentsBeta 3293
非心ベータ分布 3294
MathProbabilityDensityNoncentralBeta 3298
MathCumulativeDistributionNoncentralBeta 3300
MathQuantileNoncentralBeta 3302
MathRandomNoncentralBeta 3304
MathMomentsNoncentralBeta 3305
ガンマ分布 3306
MathProbabilityDensityGamma 3310
MathCumulativeDistributionGamma 3312
MathQuantileGamma 3314
MathRandomGamma 3316
MathMomentsGamma 3317
カイ2乗分布 3318
MathProbabilityDensityChiSquare 3322
MathCumulativeDistributionChiSquare 3324

© 2000-2025, MetaQuotes Ltd.

MathQuantileChiSquare ... 3326
MathRandomChiSquare ... 3328
MathMomentsChiSquare ... 3329
非心カイ2乗分布 ... 3330
MathProbabilityDensityNoncentralChiSquare ... 3334
MathCumulativeDistributionNoncentralChiSquare ... 3336
MathQuantileNoncentralChiSquare ... 3338
MathRandomNoncentralChiSquare ... 3340
MathMomentsNoncentralChiSquare ... 3341
指数分布 ... 3342
MathProbabilityDensityExponential ... 3346
MathCumulativeDistributionExponential ... 3348
MathQuantileExponential ... 3350
MathRandomExponential ... 3352
MathMomentsExponential ... 3353
F分布 ... 3354
MathProbabilityDensityF ... 3358
MathCumulativeDistributionF ... 3360
MathQuantileF ... 3362
MathRandomF ... 3364
MathMomentsF ... 3365
非心F分布 ... 3366
MathProbabilityDensityNoncentralF ... 3370
MathCumulativeDistributionNoncentralF ... 3372
MathQuantileNoncentralF ... 3374
MathRandomNoncentralF ... 3376
MathMomentsNoncentralF ... 3377
t分布 ... 3378
MathProbabilityDensityT ... 3382
MathCumulativeDistributionT ... 3384
MathQuantileT ... 3386
MathRandomT ... 3388
MathMomentsT ... 3389
非心t分布 ... 3390
MathProbabilityDensityNoncentralT ... 3394
MathCumulativeDistributionNoncentralT ... 3396
MathQuantileNoncentralT ... 3398
MathRandomNoncentralT ... 3400
MathMomentsNoncentralT ... 3401
ロジスティック分布 ... 3402
MathProbabilityDensityLogistic ... 3406
MathCumulativeDistributionLogistic ... 3408
MathQuantileLogistic ... 3410
MathRandomLogistic ... 3412
MathMomentsLogistic ... 3413
コーシー分布 ... 3414
MathProbabilityDensityCauchy ... 3418
MathCumulativeDistributionCauchy ... 3420
MathQuantileCauchy ... 3422
MathRandomCauchy ... 3424
MathMomentsCauchy ... 3425
一様分布 ... 3426
MathProbabilityDensityUniform ... 3430
MathCumulativeDistributionUniform ... 3432
MathQuantileUniform ... 3434
MathRandomUniform ... 3436
MathMomentsUniform ... 3437
ワイブル分布 ... 3438

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityWeibull ... 3442
MathCumulativeDistributionWeibull ... 3444
MathQuantileWeibull ... 3446
MathRandomWeibull ... 3448
MathMomentsWeibull ... 3449
二項分布 ... 3450
MathProbabilityDensityBinomial ... 3453
MathCumulativeDistributionBinomial ... 3455
MathQuantileBinomial ... 3457
MathRandomBinomial ... 3459
MathMomentsBinomial ... 3460
負の二項分布 ... 3461
MathProbabilityDensityNegativeBinomial ... 3464
MathCumulativeDistributionNegativeBinomial ... 3466
MathQuantileNegativeBinomial ... 3468
MathRandomNegativeBinomial ... 3470
MathMomentsNegativeBinomial ... 3471
幾何分布 ... 3472
MathProbabilityDensityGeometric ... 3476
MathCumulativeDistributionGeometric ... 3478
MathQuantileGeometric ... 3480
MathRandomGeometric ... 3482
MathMomentsGeometric ... 3483
超幾何分布 ... 3484
MathProbabilityDensityHypergeometric ... 3488
MathCumulativeDistributionHypergeometric ... 3490
MathQuantileHypergeometric ... 3492
MathRandomHypergeometric ... 3494
MathMomentsHypergeometric ... 3495
ポアソン分布 ... 3496
MathProbabilityDensityPoisson ... 3500
MathCumulativeDistributionPoisson ... 3502
MathQuantilePoisson ... 3504
MathRandomPoisson ... 3506
MathMomentsPoisson ... 3507
サブ関数 ... 3508
MathRandomNonZero ... 3512
MathMoments ... 3513
MathPowInt ... 3514
MathFactorial ... 3515
MathTrunc ... 3516
MathRound ... 3517
MathArctan2 ... 3519
MathGamma ... 3521
MathGammaLog ... 3522
MathBeta ... 3523
MathBetaLog ... 3524
MathBetaIncomplete ... 3525
MathGammaIncomplete ... 3526
MathBinomialCoefficient ... 3527
MathBinomialCoefficientLog ... 3528
MathHypergeometric2F2 ... 3529
MathSequence ... 3530
MathSequenceByCount ... 3531
MathReplicate ... 3532
MathReverse ... 3533
MathIdentical ... 3534
MathUnique ... 3535

© 2000-2025, MetaQuotes Ltd.

MathQuickSortAscending ... 3536
MathQuickSortDescending ... 3537
MathQuickSort ... 3538
MathOrder ... 3539
MathBitwiseNot ... 3540
MathBitwiseAnd ... 3541
MathBitwiseOr ... 3542
MathBitwiseXor ... 3543
MathBitwiseShiftL ... 3544
MathBitwiseShiftR ... 3545
MathCumulativeSum ... 3546
MathCumulativeProduct ... 3547
MathCumulativeMin ... 3548
MathCumulativeMax ... 3549
MathSin ... 3550
MathCos ... 3551
MathTan ... 3552
MathArcsin ... 3553
MathArccos ... 3554
MathArctan ... 3555
MathSinPi ... 3556
MathCosPi ... 3557
MathTanPi ... 3558
MathAbs ... 3559
MathCeil ... 3560
MathFloor ... 3561
MathSqrt ... 3562
MathExp ... 3563
MathPow ... 3564
MathLog ... 3565
MathLog2 ... 3566
MathLog10 ... 3567
MathLog1p ... 3568
MathDifference ... 3569
MathSample ... 3571
MathTukeySummary ... 3573
MathRange ... 3574
MathMin ... 3575
MathMax ... 3576
MathSum ... 3577
MathProduct ... 3578
MathStandardDeviation ... 3579
MathAverageDeviation ... 3580
MathMedian ... 3581
MathMean ... 3582
MathVariance ... 3583
MathSkewness ... 3584
MathKurtosis ... 3585
MathExpm1 ... 3586
MathSinh ... 3587
MathCosh ... 3588
MathTanh ... 3589
MathArcsinh ... 3590
MathArccosh ... 3591
MathArctanh ... 3592
MathSignif ... 3593
MathRank ... 3595
MathCorrelationPearson ... 3596

© 2000-2025, MetaQuotes Ltd.

MathCorrelationSpearman ... 3597
MathCorrelationKendall ... 3598
MathQuantile ... 3599
MathProbabilityDensityEmpirical ... 3600
MathCumulativeDistributionEmpirical ... 3601
ファジー論理 ... 3602
メンバーシップ関数 ... 3603
CConstantMembershipFunction ... 3604
GetValue ... 3606
CCompositeMembershipFunction ... 3607
CompositionType ... 3609
MembershipFunctions ... 3609
GetValue ... 3609
CDifferencTwoSigmoidalMembershipFunction ... 3611
A1 ... 3613
A2 ... 3613
C1 ... 3614
C2 ... 3614
GetValue ... 3615
CGeneralizedBellShapedMembershipFunction ... 3616
A ... 3618
B ... 3618
C ... 3619
GetValue ... 3619
CNormalCombinationMembershipFunction ... 3620
B1 ... 3622
B2 ... 3622
Sigma1 ... 3623
Sigma2 ... 3623
GetValue ... 3624
CNormalMembershipFunction ... 3625
B ... 3627
Sigma ... 3627
GetValue ... 3627
CP_ShapedMembershipFunction ... 3629
A ... 3631
B ... 3631
C ... 3632
D ... 3632
GetValue ... 3632
CProductTwoSigmoidalMembershipFunctions ... 3634
A1 ... 3636
A2 ... 3636
C1 ... 3637
C2 ... 3637
GetValue ... 3638
CS_ShapedMembershipFunction ... 3639
A ... 3641
B ... 3641
GetValue ... 3642
CSigmoidalMembershipFunction ... 3643
A ... 3645
C ... 3645
GetValue ... 3646
CTrapezoidMembershipFunction ... 3647
X1 ... 3649
X2 ... 3649
X3 ... 3650

© 2000-2025, MetaQuotes Ltd.

X4 ... 3650
GetValue ... 3651
CTriangularMembershipFunction ... 3652
X1 ... 3654
X2 ... 3654
X3 ... 3655
ToNormalMF ... 3655
GetValue ... 3655
CZ_ShapedMembershipFunction ... 3657
A ... 3659
B ... 3659
GetValue ... 3660
IMembershipFunction ... 3661
GetValue ... 3661
ファジーシステムルール ... 3662
CMamdaniFuzzyRule ... 3663
Conclusion ... 3663
Weight ... 3664
CSugenoFuzzyRule ... 3665
Conclusion ... 3665
CSingleCondition ... 3667
Not ... 3667
Term ... 3668
Var ... 3668
CConditions ... 3670
ConditionsList ... 3670
Not ... 3671
Op ... 3671
CGenericFuzzyRule ... 3672
Conclusion ... 3672
Condition ... 3673
CreateCondition ... 3673
ファジーシステム変数 ... 3675
CFuzzyVariable ... 3676
AddTerm ... 3677
GetTermByName ... 3677
Max ... 3677
Min ... 3678
Terms ... 3678
Values ... 3678
CSugenoVariable ... 3680
Functions ... 3680
GetFuncByName ... 3681
Values ... 3681
ファジー用語 ... 3682
MembershipFunction ... 3683
ファジーシステム ... 3684
Mamdani system ... 3685
AggregationMethod ... 3685
Calculate ... 3686
DefuzzificationMethod ... 3686
EmptyRule ... 3686
ImplicationMethod ... 3686
Output ... 3687
OutputByName ... 3687
ParseRule ... 3687
Rules ... 3687
Sugeno system ... 3689

© 2000-2025, MetaQuotes Ltd.

Calculate ... 3689
CreateSugenoFunction ... 3690
EmptyRule ... 3691
Output ... 3691
OutputByName ... 3691
ParseRule ... 3691
Rules ... 3692
OpenCL ... 3693
BufferCreate ... 3695
BufferFree ... 3696
BufferFromArray ... 3697
BufferRead ... 3698
BufferWrite ... 3699
Execute ... 3700
GetContext ... 3701
GetKernel ... 3702
GetKernelName ... 3703
GetProgram ... 3704
Initialize ... 3705
KernelCreate ... 3706
KernelFree ... 3707
SetArgument ... 3708
SetArgumentBuffer ... 3709
SetArgumentLocalMemory ... 3710
SetBuffersCount ... 3711
SetKernelsCount ... 3712
Shutdown ... 3713
SupportDouble ... 3714
基本クラスCObject ... 3715
Prev ... 3716
Prev ... 3717
Next ... 3718
Next ... 3719
Compare ... 3720
Save ... 3722
Load ... 3724
Type ... 3726
データ収集 ... 3727
CArray ... 3728
Step ... 3730
Step ... 3731
Total ... 3732
Available ... 3733
Max ... 3734
IsSorted ... 3735
SortMode ... 3736
Clear ... 3737
Sort ... 3738
Save ... 3739
Load ... 3740
CArrayChar ... 3741
Reserve ... 3743
Resize ... 3744
Shutdown ... 3745
Add ... 3746
AddArray ... 3747
AddArray ... 3748
Insert ... 3750

© 2000-2025, MetaQuotes Ltd.

InsertArray ... 3751
InsertArray ... 3752
AssignArray ... 3754
AssignArray ... 3755
Update ... 3757
Shift ... 3758
Delete ... 3759
DeleteRange ... 3760
At ... 3761
CompareArray ... 3763
CompareArray ... 3764
InsertSort ... 3765
Search ... 3766
SearchGreat ... 3767
SearchLess ... 3768
SearchGreatOrEqual ... 3769
SearchLessOrEqual ... 3770
SearchFirst ... 3771
SearchLast ... 3772
SearchLinear ... 3773
Save ... 3774
Load ... 3775
Type ... 3777
CArrayShort ... 3778
Reserve ... 3780
Resize ... 3781
Shutdown ... 3782
Add ... 3783
AddArray ... 3784
AddArray ... 3785
Insert ... 3787
InsertArray ... 3788
InsertArray ... 3789
AssignArray ... 3791
AssignArray ... 3792
Update ... 3794
Shift ... 3795
Delete ... 3796
DeleteRange ... 3797
At ... 3798
CompareArray ... 3800
CompareArray ... 3801
InsertSort ... 3802
Search ... 3803
SearchGreat ... 3804
SearchLess ... 3805
SearchGreatOrEqual ... 3806
SearchLessOrEqual ... 3807
SearchFirst ... 3808
SearchLast ... 3809
SearchLinear ... 3810
Save ... 3811
Load ... 3813
Type ... 3815
CArrayInt ... 3816
Reserve ... 3818
Resize ... 3819
Shutdown ... 3820

© 2000-2025, MetaQuotes Ltd.

Add ... 3821
AddArray ... 3822
AddArray ... 3823
Insert ... 3825
InsertArray ... 3826
InsertArray ... 3827
AssignArray ... 3829
AssignArray ... 3830
Update ... 3832
Shift ... 3833
Delete ... 3834
DeleteRange ... 3835
At ... 3836
CompareArray ... 3838
CompareArray ... 3839
InsertSort ... 3840
Search ... 3841
SearchGreat ... 3842
SearchLess ... 3843
SearchGreatOrEqual ... 3844
SearchLessOrEqual ... 3845
SearchFirst ... 3846
SearchLast ... 3847
SearchLinear ... 3848
Save ... 3849
Load ... 3851
Type ... 3853
CArrayLong ... 3854
Reserve ... 3856
Resize ... 3857
Shutdown ... 3858
Add ... 3859
AddArray ... 3860
AddArray ... 3861
Insert ... 3863
InsertArray ... 3864
InsertArray ... 3865
AssignArray ... 3867
AssignArray ... 3868
Update ... 3870
Shift ... 3871
Delete ... 3872
DeleteRange ... 3873
At ... 3874
CompareArray ... 3876
CompareArray ... 3877
InsertSort ... 3878
Search ... 3879
SearchGreat ... 3880
SearchLess ... 3881
SearchGreatOrEqual ... 3882
SearchLessOrEqual ... 3883
SearchFirst ... 3884
SearchLast ... 3885
SearchLinear ... 3886
Save ... 3887
Load ... 3889
Type ... 3891

© 2000-2025, MetaQuotes Ltd.

CArrayFloat 3892
Delta 3894
Reserve 3895
Resize 3896
Shutdown 3897
Add 3898
AddArray 3899
AddArray 3900
Insert 3902
InsertArray 3903
InsertArray 3904
AssignArray 3906
AssignArray 3907
Update 3909
Shift 3910
Delete 3911
DeleteRange 3912
At 3913
CompareArray 3915
CompareArray 3916
InsertSort 3917
Search 3918
SearchGreat 3919
SearchLess 3920
SearchGreatOrEqual 3921
SearchLessOrEqual 3922
SearchFirst 3923
SearchLast 3924
SearchLinear 3925
Save 3926
Load 3928
Type 3930
CArrayDouble 3931
Delta 3934
Reserve 3935
Resize 3936
Shutdown 3937
Add 3938
AddArray 3939
AddArray 3940
Insert 3942
InsertArray 3943
InsertArray 3944
AssignArray 3946
AssignArray 3947
Update 3949
Shift 3950
Delete 3951
DeleteRange 3952
At 3953
CompareArray 3955
CompareArray 3956
Minimum 3957
Maximum 3958
InsertSort 3959
Search 3960
SearchGreat 3961
SearchLess 3962

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual ... 3963
SearchLessOrEqual ... 3964
SearchFirst ... 3965
SearchLast ... 3966
SearchLinear ... 3967
Save ... 3968
Load ... 3970
Type ... 3972
CArrayString ... 3973
Reserve ... 3975
Resize ... 3976
Shutdown ... 3977
Add ... 3978
AddArray ... 3979
AddArray ... 3980
Insert ... 3982
InsertArray ... 3983
InsertArray ... 3984
AssignArray ... 3986
AssignArray ... 3987
Update ... 3989
Shift ... 3990
Delete ... 3991
DeleteRange ... 3992
At ... 3993
CompareArray ... 3995
CompareArray ... 3996
InsertSort ... 3997
Search ... 3998
SearchGreat ... 3999
SearchLess ... 4000
SearchGreatOrEqual ... 4001
SearchLessOrEqual ... 4002
SearchFirst ... 4003
SearchLast ... 4004
SearchLinear ... 4005
Save ... 4006
Load ... 4008
Type ... 4010
CArrayObj ... 4011
FreeMode ... 4016
FreeMode ... 4017
Reserve ... 4019
Resize ... 4020
Clear ... 4021
Shutdown ... 4022
CreateElement ... 4023
Add ... 4025
AddArray ... 4026
Insert ... 4029
InsertArray ... 4031
AssignArray ... 4033
Update ... 4035
Shift ... 4036
Detach ... 4037
Delete ... 4038
DeleteRange ... 4039
At ... 4040

© 2000-2025, MetaQuotes Ltd.

CompareArray 4041
InsertSort 4042
Search 4043
SearchGreat 4044
SearchLess 4045
SearchGreatOrEqual 4046
SearchLessOrEqual 4047
SearchFirst 4048
SearchLast 4049
Save 4050
Load 4051
Type 4053
CList 4054
FreeMode 4056
FreeMode 4057
Total 4059
IsSorted 4060
SortMode 4061
CreateElement 4062
Add 4063
Insert 4064
DetachCurrent 4066
DeleteCurrent 4067
Delete 4068
Clear 4069
IndexOf 4070
GetNodeAtIndex 4071
GetFirstNode 4072
GetPrevNode 4073
GetCurrentNode 4074
GetNextNode 4075
GetLastNode 4076
Sort 4077
MoveToIndex 4078
Exchange 4079
CompareList 4080
Search 4081
Save 4082
Load 4084
Type 4086
CTreeNode 4087
Owner 4092
Left 4093
Right 4094
Balance 4095
BalanceL 4096
BalanceR 4097
CreateSample 4098
RefreshBalance 4099
GetNext 4100
SaveNode 4101
LoadNode 4102
Type 4103
CTree 4104
Root 4110
CreateElement 4111
Insert 4112
Detach 4113

© 2000-2025, MetaQuotes Ltd.

Delete ... 4114
Clear ... 4115
Find ... 4116
Save ... 4117
Load ... 4118
Type ... 4119
ジェネリックデータコレクション ... 4120
ICollection<T> ... 4122
Add ... 4123
Count ... 4124
Contains ... 4125
CopyTo ... 4126
Clear ... 4127
Remove ... 4128
IEqualityComparable<T> ... 4129
Equals ... 4130
HashCode ... 4131
IComparable<T> ... 4132
Compare ... 4133
IComparer<T> ... 4134
Compare ... 4135
IEqualityComparer<T> ... 4136
Equals ... 4137
HashCode ... 4138
IList<T> ... 4139
TryGetValue ... 4140
TrySetValue ... 4141
Insert ... 4142
IndexOf ... 4143
LastIndexOf ... 4144
RemoveAt ... 4145
IMap<TKey,TValue> ... 4146
Add ... 4147
Contains ... 4148
Remove ... 4149
TryGetValue ... 4150
TrySetValue ... 4151
CopyTo ... 4152
ISet<T> ... 4153
ExceptWith ... 4154
IntersectWith ... 4155
SymmetricExceptWith ... 4156
UnionWith ... 4157
IsProperSubsetOf ... 4158
IsProperSupersetOf ... 4159
IsSubsetOf ... 4160
IsSupersetOf ... 4161
Overlaps ... 4162
SetEquals ... 4163
CDefaultComparer<T> ... 4164
Compare ... 4165
CDefaultEqualityComparer<T> ... 4166
Equals ... 4167
HashCode ... 4168
CRedBlackTreeNode<T> ... 4169
Value ... 4170
Parent ... 4171
Left ... 4172

© 2000-2025, MetaQuotes Ltd.

Right ... 4173
Color ... 4174
IsLeaf ... 4175
CreateEmptyNode ... 4176
CLinkedListNode<T> ... 4177
List ... 4178
Next ... 4179
Previous ... 4180
Value ... 4181
CKeyValuePair<TKey,TValue> ... 4182
Key ... 4183
Value ... 4184
Clone ... 4185
Compare ... 4186
Equals ... 4187
HashCode ... 4188
CArrayList<T> ... 4189
Capacity ... 4191
Count ... 4192
Contains ... 4193
TrimExcess ... 4194
TryGetValue ... 4195
TrySetValue ... 4196
Add ... 4197
AddRange ... 4198
Insert ... 4199
InsertRange ... 4200
CopyTo ... 4201
BinarySearch ... 4202
IndexOf ... 4203
LastIndexOf ... 4204
Clear ... 4205
Remove ... 4206
RemoveAt ... 4207
RemoveRange ... 4208
Reverse ... 4209
Sort ... 4210
CHashMap<TKey,TValue> ... 4211
Add ... 4212
Count ... 4213
Comparer ... 4214
Contains ... 4215
ContainsKey ... 4216
ContainsValue ... 4217
CopyTo ... 4218
Clear ... 4219
Remove ... 4220
TryGetValue ... 4221
TrySetValue ... 4222
CHashSet<T> ... 4223
Add ... 4225
Count ... 4226
Contains ... 4227
Comparer ... 4228
TrimExcess ... 4229
CopyTo ... 4230
Clear ... 4231
Remove ... 4232

© 2000-2025, MetaQuotes Ltd.

ExceptWith ... 4233
IntersectWith ... 4234
SymmetricExceptWith ... 4235
UnionWith ... 4236
IsProperSubsetOf ... 4237
IsProperSupersetOf ... 4238
IsSubsetOf ... 4239
IsSupersetOf ... 4240
Overlaps ... 4241
SetEquals ... 4242
CLinkedList<T> ... 4243
Add ... 4245
AddAfter ... 4246
AddBefore ... 4247
AddFirst ... 4248
AddLast ... 4249
Count ... 4250
Head ... 4251
First ... 4252
Last ... 4253
Contains ... 4254
CopyTo ... 4255
Clear ... 4256
Remove ... 4257
RemoveFirst ... 4258
RemoveLast ... 4259
Find ... 4260
FindLast ... 4261
CQueue<T> ... 4262
Add ... 4263
Enqueue ... 4264
Count ... 4265
Contains ... 4266
TrimExcess ... 4267
CopyTo ... 4268
Clear ... 4269
Remove ... 4270
Dequeue ... 4271
Peek ... 4272
CRedBlackTree<T> ... 4273
Add ... 4275
Count ... 4276
Root ... 4277
Contains ... 4278
Comparer ... 4279
TryGetMin ... 4280
TryGetMax ... 4281
CopyTo ... 4282
Clear ... 4283
Remove ... 4284
RemoveMin ... 4285
RemoveMax ... 4286
Find ... 4287
FindMin ... 4288
FindMax ... 4289
CSortedMap<TKey, TValue> ... 4290
Add ... 4291
Count ... 4292

© 2000-2025, MetaQuotes Ltd.

Comparer ... 4293
Contains ... 4294
ContainsKey ... 4295
ContainsValue ... 4296
CopyTo ... 4297
Clear ... 4298
Remove ... 4299
TryGetValue ... 4300
TrySetValue ... 4301
CSortedSet<T> ... 4302
Add ... 4304
Count ... 4305
Contains ... 4306
Comparer ... 4307
TryGetMin ... 4308
TryGetMax ... 4309
CopyTo ... 4310
Clear ... 4311
Remove ... 4312
ExceptWith ... 4313
IntersectWith ... 4314
SymmetricExceptWith ... 4315
UnionWith ... 4316
IsProperSubsetOf ... 4317
IsProperSupersetOf ... 4318
IsSubsetOf ... 4319
IsSupersetOf ... 4320
Overlaps ... 4321
SetEquals ... 4322
GetViewBetween ... 4323
GetReverse ... 4324
CStack<T> ... 4325
Add ... 4326
Count ... 4327
Contains ... 4328
TrimExcess ... 4329
CopyTo ... 4330
Clear ... 4331
Remove ... 4332
Push ... 4333
Peek ... 4334
Pop ... 4335
ArrayBinarySearch<T> ... 4336
ArrayIndexOf<T> ... 4337
ArrayLastIndexOf<T> ... 4338
ArrayReverse<T> ... 4339
Compare ... 4340
Equals<T> ... 4343
GetHashCode ... 4344
ファイル ... 4347
CFile ... 4348
Handle ... 4350
Filename ... 4351
Flags ... 4352
SetUnicode ... 4353
SetCommon ... 4354
Open ... 4355
Close ... 4356

© 2000-2025, MetaQuotes Ltd.

Delete 4357
IsExist 4358
Copy 4359
Move 4360
Size 4361
Tell 4362
Seek 4363
Flush 4364
IsEnding 4365
IsLineEnding 4366
FolderCreate 4367
FolderDelete 4368
FolderClean 4369
FileFindFirst 4370
FileFindNext 4371
FileFindClose 4372
CFileBin 4373
Open 4375
WriteChar 4376
WriteShort 4377
WriteInteger 4378
WriteLong 4379
WriteFloat 4380
WriteDouble 4381
WriteString 4382
WriteCharArray 4383
WriteShortArray 4384
WriteIntegerArray 4385
WriteLongArray 4386
WriteFloatArray 4387
WriteDoubleArray 4388
WriteObject 4389
ReadChar 4390
ReadShort 4391
ReadInteger 4392
ReadLong 4393
ReadFloat 4394
ReadDouble 4395
ReadString 4396
ReadCharArray 4397
ReadShortArray 4398
ReadIntegerArray 4399
ReadLongArray 4400
ReadFloatArray 4401
ReadDoubleArray 4402
ReadObject 4403
CFileTxt 4404
Open 4405
WriteString 4406
ReadString 4407
文字列 4408
CString 4409
Str 4411
Len 4412
Copy 4413
Fill 4414
Assign 4415
Append 4416

© 2000-2025, MetaQuotes Ltd.

Insert 4417
Compare 4418
CompareNoCase 4419
Left 4420
Right 4421
Mid 4422
Trim 4423
TrimLeft 4424
TrimRight 4425
Clear 4426
ToUpper 4427
ToLower 4428
Reverse 4429
Find 4430
FindRev 4431
Remove 4432
Replace 4433
グラフィックオブジェクト 4434
CChartObject 4435
ChartId 4438
Window 4439
Name 4440
NumPoints 4441
Attach 4442
SetPoint 4443
Delete 4444
Detach 4445
ShiftObject 4446
ShiftPoint 4447
Time 4448
Price 4450
Color 4452
Style 4453
Width 4454
Background 4455
Selected 4456
Selectable 4457
Description 4458
Tooltip 4459
Timeframes 4460
Z_Order 4461
CreateTime 4462
LevelsCount 4463
LevelColor 4464
LevelStyle 4466
LevelWidth 4468
LevelValue 4470
LevelDescription 4472
GetInteger 4474
SetInteger 4476
GetDouble 4478
SetDouble 4480
GetString 4482
SetString 4484
Save 4486
Load 4487
Type 4488
ラインオブジェクト 4489

© 2000-2025, MetaQuotes Ltd.

CChartObjectVLine ... 4490
Create ... 4491
Type ... 4492
CChartObjectHLine ... 4493
Create ... 4494
Type ... 4495
CChartObjectTrend ... 4496
Create ... 4498
RayLeft ... 4499
RayRight ... 4500
Save ... 4501
Load ... 4502
Type ... 4503
CChartObjectTrendByAngle ... 4504
Create ... 4506
Angle ... 4507
Type ... 4508
CChartObjectCycles ... 4509
Create ... 4510
Type ... 4511
チャンネルオブジェクト ... 4512
CChartObjectChannel ... 4513
Create ... 4514
Type ... 4515
CChartObjectRegression ... 4516
Create ... 4518
Type ... 4519
CChartObjectStdDevChannel ... 4520
Create ... 4522
Deviations ... 4523
Save ... 4524
Load ... 4525
Type ... 4526
CChartObjectPitchfork ... 4527
Create ... 4529
Type ... 4530
ギャンツール ... 4531
CChartObjectGannLine ... 4532
Create ... 4534
PipsPerBar ... 4535
Save ... 4536
Load ... 4537
Type ... 4538
CChartObjectGannFan ... 4539
Create ... 4541
PipsPerBar ... 4542
Downtrend ... 4543
Save ... 4544
Load ... 4545
Type ... 4546
CChartObjectGannGrid ... 4547
Create ... 4549
PipsPerBar ... 4550
Downtrend ... 4551
Save ... 4552
Load ... 4553
Type ... 4554
フィボナッチツール ... 4555

© 2000-2025, MetaQuotes Ltd.

CChartObjectFibo ... 4556
Create ... 4557
Type ... 4558
CChartObjectFiboTimes ... 4559
Create ... 4560
Type ... 4561
CChartObjectFiboFan ... 4562
Create ... 4563
Type ... 4564
CChartObjectFiboArc ... 4565
Create ... 4567
Scale ... 4568
Ellipse ... 4569
Save ... 4570
Load ... 4571
Type ... 4572
CChartObjectFiboChannel ... 4573
Create ... 4575
Type ... 4576
CChartObjectFiboExpansion ... 4577
Create ... 4579
Type ... 4580
エリオットツール ... 4581
CChartObjectElliottWave3 ... 4582
Create ... 4584
Degree ... 4585
Lines ... 4586
Save ... 4587
Load ... 4588
Type ... 4589
CChartObjectElliottWave5 ... 4590
Create ... 4591
Type ... 4593
図形オブジェクト ... 4594
CChartObjectRectangle ... 4595
Create ... 4596
Type ... 4597
CChartObjectTriangle ... 4598
Create ... 4599
Type ... 4600
CChartObjectEllipse ... 4601
Create ... 4602
Type ... 4603
矢印オブジェクト ... 4604
CChartObjectArrow ... 4605
Create ... 4607
ArrowCode ... 4609
Anchor ... 4611
Save ... 4613
Load ... 4614
Type ... 4615
固定コードを持つ矢印 ... 4616
Create ... 4618
ArrowCode ... 4620
Type ... 4621
オブジェクト制御 ... 4622
CChartObjectText ... 4623
Create ... 4625

© 2000-2025, MetaQuotes Ltd.

Angle ... 4626
Font ... 4627
FontSize ... 4628
Anchor ... 4629
Save ... 4630
Load ... 4631
Type ... 4632
CChartObjectLabel ... 4633
Create ... 4635
X_Distance ... 4636
Y_Distance ... 4637
X_Size ... 4638
Y_Size ... 4639
Corner ... 4640
Time ... 4641
Price ... 4642
Save ... 4643
Load ... 4644
Type ... 4645
CChartObjectEdit ... 4646
Create ... 4648
TextAlign ... 4649
X_Size ... 4650
Y_Size ... 4651
BackColor ... 4652
BorderColor ... 4653
ReadOnly ... 4654
Angle ... 4655
Save ... 4656
Load ... 4657
Type ... 4658
CChartObjectButton ... 4659
State ... 4661
Save ... 4662
Load ... 4663
Type ... 4664
CChartObjectSubChart ... 4665
Create ... 4667
X_Distance ... 4668
Y_Distance ... 4669
Corner ... 4670
X_Size ... 4671
Y_Size ... 4672
Symbol ... 4673
Period ... 4674
Scale ... 4675
DateScale ... 4676
PriceScale ... 4677
Time ... 4678
Price ... 4679
Save ... 4680
Load ... 4681
Type ... 4682
CChartObjectBitmap ... 4683
Create ... 4685
BmpFile ... 4686
X_Offset ... 4687
Y_Offset ... 4688

© 2000-2025, MetaQuotes Ltd.

Save ... 4689
Load ... 4690
Type ... 4691
CChartObjectBmpLabel ... 4692
Create ... 4694
X_Distance ... 4695
Y_Distance ... 4696
X_Offset ... 4697
Y_Offset ... 4698
Corner ... 4699
X_Size ... 4700
Y_Size ... 4701
BmpFileOn ... 4702
BmpFileOff ... 4703
State ... 4704
Time ... 4705
Price ... 4706
Save ... 4707
Load ... 4708
Type ... 4709
CChartObjectRectLabel ... 4710
Create ... 4712
X_Size ... 4713
Y_Size ... 4714
BackColor ... 4715
Angle ... 4716
BorderType ... 4717
Save ... 4718
Load ... 4719
Type ... 4720
カスタムグラフィックス ... 4721
CCanvas ... 4722
Attach ... 4726
Arc ... 4727
Pie ... 4730
FillPolygon ... 4733
FillEllipse ... 4734
GetDefaultColor ... 4735
ChartObjectName ... 4736
Circle ... 4737
CircleAA ... 4738
CircleWu ... 4739
Create ... 4740
CreateBitmap ... 4741
CreateBitmapLabel ... 4743
Destroy ... 4745
Ellipse ... 4746
EllipseAA ... 4747
EllipseWu ... 4748
Erase ... 4749
Fill ... 4750
FillCircle ... 4751
FillRectangle ... 4752
FillTriangle ... 4753
FontAngleGet ... 4754
FontAngleSet ... 4755
FontFlagsGet ... 4756
FontFlagsSet ... 4757

© 2000-2025, MetaQuotes Ltd.

FontGet ... 4758
FontNameGet ... 4759
FontNameSet ... 4760
FontSet ... 4761
FontSizeGet ... 4762
FontSizeSet ... 4763
Height ... 4764
Line ... 4765
LineAA ... 4766
LineWu ... 4767
LineHorizontal ... 4768
LineVertical ... 4769
LineStyleSet ... 4770
LineThick ... 4771
LineThickVertical ... 4772
LineThickHorizontal ... 4773
LoadFromFile ... 4774
PixelGet ... 4775
PixelSet ... 4776
PixelSetAA ... 4777
Polygon ... 4778
PolygonAA ... 4779
PolygonWu ... 4780
PolygonThick ... 4781
PolygonSmooth ... 4782
Polyline ... 4783
PolylineSmooth ... 4784
PolylineThick ... 4785
PolylineWu ... 4786
PolylineAA ... 4787
Rectangle ... 4788
Resize ... 4789
ResourceName ... 4790
TextHeight ... 4791
TextOut ... 4792
TextSize ... 4793
TextWidth ... 4794
TransparentLevelSet ... 4795
Triangle ... 4796
TriangleAA ... 4797
TriangleWu ... 4798
Update ... 4799
Width ... 4800
CChartCanvas ... 4801
ColorBackground ... 4804
ColorBorder ... 4805
ColorText ... 4806
ColorGrid ... 4807
MaxData ... 4808
MaxDescrLen ... 4809
ShowFlags ... 4810
IsShowLegend ... 4811
IsShowScaleLeft ... 4812
IsShowScaleRight ... 4813
IsShowScaleTop ... 4814
IsShowScaleBottom ... 4815
IsShowGrid ... 4816
IsShowDescriptors ... 4817

© 2000-2025, MetaQuotes Ltd.

IsShowPercent 4818
VScaleMin 4819
VScaleMax 4820
NumGrid 4821
DataOffset 4822
DataTotal 4823
DrawDescriptors 4824
DrawData 4825
Create 4826
AllowedShowFlags 4827
ShowLegend 4828
ShowScaleLeft 4829
ShowScaleRight 4830
ShowScaleTop 4831
ShowScaleBottom 4832
ShowGrid 4833
ShowDescriptors 4834
ShowValue 4835
ShowPercent 4836
LegendAlignment 4837
Accumulative 4838
VScaleParams 4839
DescriptorUpdate 4840
ColorUpdate 4841
ValuesCheck 4842
Redraw 4843
DrawBackground 4844
DrawLegend 4845
DrawLegendVertical 4846
DrawLegendHorizontal 4847
CalcScales 4848
DrawScales 4849
DrawScaleLeft 4850
DrawScaleRight 4851
DrawScaleTop 4852
DrawScaleBottom 4853
DrawGrid 4854
DrawChart 4855
CHistogramChart 4856
Gradient 4861
BarGap 4862
BarMinSize 4863
BarBorder 4864
Create 4865
SeriesAdd 4866
SeriesInsert 4867
SeriesUpdate 4868
SeriesDelete 4869
ValueUpdate 4870
DrawData 4871
DrawBar 4872
GradientBrush 4873
CLineChart 4874
Filled 4878
Create 4879
SeriesAdd 4880
SeriesInsert 4881
SeriesUpdate 4882

© 2000-2025, MetaQuotes Ltd.

SeriesDelete 4883
ValueUpdate 4884
DrawChart 4885
DrawData 4886
CalcArea 4887
CPieChart 4888
Create 4892
SeriesSet 4893
ValueAdd 4894
ValueInsert 4895
ValueUpdate 4896
ValueDelete 4897
DrawChart 4898
DrawPie 4899
LabelMake 4900
3Dグラフィックス 4901
CCanvas3D 4902
AmbientColorGet 4904
AmbientColorSet 4905
Attach 4906
Create 4907
Destroy 4908
DXContext 4909
DXDispatcher 4910
InputScene 4911
LightColorGet 4912
LightColorSet 4913
LightDirectionGet 4914
LightDirectionSet 4915
ObjectAdd 4916
ProjectionMatrixGet 4917
ProjectionMatrixSet 4918
Render 4919
RenderBegin 4920
RenderEnd 4921
ViewMatrixGet 4922
ViewMatrixSet 4923
ViewPositionSet 4924
ViewRotationSet 4925
ViewTargetSet 4926
ViewUpDirectionSet 4927
価格チャート 4928
ChartID 4932
Mode 4933
Foreground 4934
Shift 4935
ShiftSize 4936
AutoScroll 4937
Scale 4938
ScaleFix 4939
ScaleFix_11 4940
FixedMax 4941
FixedMin 4942
PointsPerBar 4943
ScalePPB 4944
ShowOHLC 4945
ShowLineBid 4946
ShowLineAsk 4947

© 2000-2025, MetaQuotes Ltd.

ShowLastLine 4948
ShowPeriodSep 4949
ShowGrid 4950
ShowVolumes 4951
ShowObjectDescr 4952
ShowDateScale 4953
ShowPriceScale 4954
ColorBackground 4955
ColorForeground 4956
ColorGrid 4957
ColorBarUp 4958
ColorBarDown 4959
ColorCandleBull 4960
ColorCandleBear 4961
ColorChartLine 4962
ColorVolumes 4963
ColorLineBid 4964
ColorLineAsk 4965
ColorLineLast 4966
ColorStopLevels 4967
VisibleBars 4968
WindowsTotal 4969
WindowIsVisible 4970
WindowHandle 4971
FirstVisibleBar 4972
WidthInBars 4973
WidthInPixels 4974
HeightInPixels 4975
PriceMin 4976
PriceMax 4977
Attach 4978
FirstChart 4979
NextChart 4980
Open 4981
Detach 4982
Close 4983
BringToTop 4984
EventObjectCreate 4985
EventObjectDelete 4986
IndicatorAdd 4987
IndicatorDelete 4988
IndicatorsTotal 4989
IndicatorName 4990
Navigate 4991
Symbol 4992
Period 4993
Redraw 4994
GetInteger 4995
SetInteger 4996
GetDouble 4997
SetDouble 4998
GetString 4999
SetString 5000
SetSymbolPeriod 5001
ApplyTemplate 5002
ScreenShot 5003
WindowOnDropped 5004
PriceOnDropped 5005

© 2000-2025, MetaQuotes Ltd.

TimeOnDropped 5006
XOnDropped 5007
YOnDropped 5008
Save 5009
Load 5010
Type 5011
科学的チャート 5012
GraphPlot 5013
CAxis 5017
AutoScale 5019
Min 5020
Max 5021
Step 5022
Name 5023
Color 5024
ValuesSize 5025
ValuesWidth 5026
ValuesFormat 5027
ValuesDateTimeMode 5028
ValuesFunctionFormat 5029
ValuesFunctionFormatCBData 5031
NameSize 5032
ZeroLever 5033
DefaultStep 5034
MaxLabels 5035
MinGrace 5036
MaxGrace 5037
SelectAxisScale 5038
CColorGenerator 5039
Next 5040
Reset 5041
CCurve 5042
Type 5044
Name 5045
Color 5046
XMax 5047
XMin 5048
YMax 5049
YMin 5050
Size 5051
PointsSize 5052
PointsFill 5053
PointsColor 5054
GetX 5055
GetY 5056
LinesStyle 5057
LinesIsSmooth 5058
LinesSmoothTension 5059
LinesSmoothStep 5060
LinesEndStyle 5061
LinesWidth 5062
HistogramWidth 5064
CustomPlotCBData 5065
CustomPlotFunction 5066
PointsType 5070
StepsDimension 5071
TrendLineCoefficients 5072
TrendLineColor 5073

© 2000-2025, MetaQuotes Ltd.

TrendLineVisible 5074
Update 5076
Visible 5078
CGraphic 5079
Create 5082
Destroy 5083
Update 5084
ChartObjectName 5085
ResourceName 5086
XAxis 5087
YAxis 5088
GapSize 5089
BackgroundColor 5090
BackgroundMain 5091
BackgroundMainSize 5092
BackgroundMainColor 5093
BackgroundSub 5094
BackgroundSubSize 5095
BackgroundSubColor 5096
GridLineColor 5097
GridBackgroundColor 5098
GridCircleRadius 5099
GridCircleColor 5100
GridHasCircle 5101
GridAxisLineColor 5102
HistoryNameWidth 5103
HistoryNameSize 5104
HistorySymbolSize 5105
TextAdd 5106
LineAdd 5107
CurveAdd 5108
CurvePlot 5111
CurvePlotAll 5112
CurveGetByIndex 5113
CurveGetByName 5114
CurvesTotal 5115
CurveRemoveByName 5116
CurveRemoveByIndex 5117
MarksToAxisAdd 5118
MajorMarkSize 5119
FontSet 5120
FontGet 5121
Attach 5122
CalculateMaxMinValues 5123
Height 5124
IndentDown 5125
IndentLeft 5126
IndentRight 5127
IndentUp 5128
Redraw 5129
ResetParameters 5130
ScaleX 5131
ScaleY 5132
SetDefaultParameters 5133
Width 5134
指標 5135
基本クラス 5136
CSpreadBuffer 5137

© 2000-2025, MetaQuotes Ltd.

Size ... 5139
SetSymbolPeriod ... 5140
At ... 5141
Refresh ... 5142
RefreshCurrent ... 5143
CTimeBuffer ... 5144
Size ... 5146
SetSymbolPeriod ... 5147
At ... 5148
Refresh ... 5149
RefreshCurrent ... 5150
CTickVolumeBuffer ... 5151
Size ... 5153
SetSymbolPeriod ... 5154
At ... 5155
Refresh ... 5156
RefreshCurrent ... 5157
CRealVolumeBuffer ... 5158
Size ... 5160
SetSymbolPeriod ... 5161
At ... 5162
Refresh ... 5163
RefreshCurrent ... 5164
CDoubleBuffer ... 5165
Size ... 5167
SetSymbolPeriod ... 5168
At ... 5169
Refresh ... 5170
RefreshCurrent ... 5171
COpenBuffer ... 5172
Refresh ... 5173
RefreshCurrent ... 5174
CHighBuffer ... 5175
Refresh ... 5176
RefreshCurrent ... 5177
CLowBuffer ... 5178
Refresh ... 5179
RefreshCurrent ... 5180
CCloseBuffer ... 5181
Refresh ... 5182
RefreshCurrent ... 5183
CIndicatorBuffer ... 5184
Offset ... 5186
Name ... 5187
At ... 5188
Refresh ... 5189
RefreshCurrent ... 5190
CSeries ... 5191
Name ... 5193
BuffersTotal ... 5194
Timeframe ... 5195
Symbol ... 5196
Period ... 5197
RefreshCurrent ... 5198
BufferSize ... 5199
BufferResize ... 5200
Refresh ... 5201
PeriodDescription ... 5202

© 2000-2025, MetaQuotes Ltd.

CPriceSeries ... 5203
BufferResize ... 5205
GetData ... 5206
Refresh ... 5207
MinIndex ... 5208
MinValue ... 5209
MaxIndex ... 5210
MaxValue ... 5211
CIndicator ... 5212
Handle ... 5214
Status ... 5215
FullRelease ... 5216
Create ... 5217
BufferResize ... 5218
BarsCalculated ... 5219
GetData ... 5220
Refresh ... 5223
Minimum ... 5224
MinValue ... 5225
Maximum ... 5226
MaxValue ... 5227
MethodDescription ... 5228
PriceDescription ... 5229
VolumeDescription ... 5230
AddToChart ... 5231
DeleteFromChart ... 5232
CIndicators ... 5233
Create ... 5234
Refresh ... 5235
時系列クラス ... 5236
CiSpread ... 5237
Create ... 5239
BufferResize ... 5240
GetData ... 5241
Refresh ... 5243
CiTime ... 5244
Create ... 5246
BufferResize ... 5247
GetData ... 5248
Refresh ... 5250
CiTickVolume ... 5251
Create ... 5253
BufferResize ... 5254
GetData ... 5255
Refresh ... 5257
CiRealVolume ... 5258
Create ... 5260
BufferResize ... 5261
GetData ... 5262
Refresh ... 5264
CiOpen ... 5265
Create ... 5267
GetData ... 5268
CiHigh ... 5270
Create ... 5272
GetData ... 5273
CiLow ... 5275
Create ... 5277

© 2000-2025, MetaQuotes Ltd.

GetData ... 5278
CiClose ... 5280
Create ... 5282
GetData ... 5283
トレンド指標 ... 5285
CiADX ... 5286
MaPeriod ... 5288
Create ... 5289
Main ... 5290
Plus ... 5291
Minus ... 5292
Type ... 5293
CiADXWilder ... 5294
MaPeriod ... 5296
Create ... 5297
Main ... 5298
Plus ... 5299
Minus ... 5300
Type ... 5301
CiBands ... 5302
MaPeriod ... 5304
MaShift ... 5305
Deviation ... 5306
Applied ... 5307
Create ... 5308
Base ... 5309
Upper ... 5310
Lower ... 5311
Type ... 5312
CiEnvelopes ... 5313
MaPeriod ... 5315
MaShift ... 5316
MaMethod ... 5317
Deviation ... 5318
Applied ... 5319
Create ... 5320
Upper ... 5321
Lower ... 5322
Type ... 5323
CiIchimoku ... 5324
TenkanSenPeriod ... 5326
KijunSenPeriod ... 5327
SenkouSpanBPeriod ... 5328
Create ... 5329
TenkanSen ... 5330
KijunSen ... 5331
SenkouSpanA ... 5332
SenkouSpanB ... 5333
ChinkouSpan ... 5334
Type ... 5335
CiMA ... 5336
MaPeriod ... 5338
MaShift ... 5339
MaMethod ... 5340
Applied ... 5341
Create ... 5342
Main ... 5343
Type ... 5344

© 2000-2025, MetaQuotes Ltd.

CiSAR 5345
SarStep 5347
Maximum 5348
Create 5349
Main 5350
Type 5351
CiStdDev 5352
MaPeriod 5354
MaShift 5355
MaMethod 5356
Applied 5357
Create 5358
Main 5359
Type 5360
CiDEMA 5361
MaPeriod 5363
IndShift 5364
Applied 5365
Create 5366
Main 5367
Type 5368
CiTEMA 5369
MaPeriod 5371
IndShift 5372
Applied 5373
Create 5374
Main 5375
Type 5376
CiFrAMA 5377
MaPeriod 5379
IndShift 5380
Applied 5381
Create 5382
Main 5383
Type 5384
CiAMA 5385
MaPeriod 5387
FastEmaPeriod 5388
SlowEmaPeriod 5389
IndShift 5390
Applied 5391
Create 5392
Main 5393
Type 5394
CiVIDyA 5395
CmoPeriod 5397
EmaPeriod 5398
IndShift 5399
Applied 5400
Create 5401
Main 5402
Type 5403
オシレーター 5404
CiATR 5405
MaPeriod 5407
Create 5408
Main 5409
Type 5410

© 2000-2025, MetaQuotes Ltd.

CiBearsPower ... 5411
MaPeriod ... 5413
Create ... 5414
Main ... 5415
Type ... 5416
CiBullsPower ... 5417
MaPeriod ... 5419
Create ... 5420
Main ... 5421
Type ... 5422
CiCCI ... 5423
MaPeriod ... 5425
Applied ... 5426
Create ... 5427
Main ... 5428
Type ... 5429
CiChaikin ... 5430
FastMaPeriod ... 5432
SlowMaPeriod ... 5433
MaMethod ... 5434
Applied ... 5435
Create ... 5436
Main ... 5437
Type ... 5438
CiDeMarker ... 5439
MaPeriod ... 5441
Create ... 5442
Main ... 5443
Type ... 5444
CiForce ... 5445
MaPeriod ... 5447
MaMethod ... 5448
Applied ... 5449
Create ... 5450
Main ... 5451
Type ... 5452
CiMACD ... 5453
FastEmaPeriod ... 5455
SlowEmaPeriod ... 5456
SignalPeriod ... 5457
Applied ... 5458
Create ... 5459
Main ... 5460
Signal ... 5461
Type ... 5462
CiMomentum ... 5463
MaPeriod ... 5465
Applied ... 5466
Create ... 5467
Main ... 5468
Type ... 5469
CiOsMA ... 5470
FastEmaPeriod ... 5472
SlowEmaPeriod ... 5473
SignalPeriod ... 5474
Applied ... 5475
Create ... 5476
Main ... 5477

© 2000-2025, MetaQuotes Ltd.

Type ... 5478
CiRSI ... 5479
MaPeriod ... 5481
Applied ... 5482
Create ... 5483
Main ... 5484
Type ... 5485
CiRVI ... 5486
MaPeriod ... 5488
Create ... 5489
Main ... 5490
Signal ... 5491
Type ... 5492
CiStochastic ... 5493
Kperiod ... 5495
Dperiod ... 5496
Slowing ... 5497
MaMethod ... 5498
PriceField ... 5499
Create ... 5500
Main ... 5501
Signal ... 5502
Type ... 5503
CiTriX ... 5504
MaPeriod ... 5506
Applied ... 5507
Create ... 5508
Main ... 5509
Type ... 5510
CiWPR ... 5511
CalcPeriod ... 5513
Create ... 5514
Main ... 5515
Type ... 5516
ボリューム指標 ... 5517
CiAD ... 5518
Applied ... 5520
Create ... 5521
Main ... 5522
Type ... 5523
CiMFI ... 5524
MaPeriod ... 5526
Applied ... 5527
Create ... 5528
Main ... 5529
Type ... 5530
CiOBV ... 5531
Applied ... 5533
Create ... 5534
Main ... 5535
Type ... 5536
CiVolumes ... 5537
Applied ... 5539
Create ... 5540
Main ... 5541
Type ... 5542
ビルウィリアムズ指標 ... 5543
CiAC ... 5544

© 2000-2025, MetaQuotes Ltd.

Create ... 5546
Main ... 5547
Type ... 5548
CiAlligator ... 5549
JawPeriod ... 5551
JawShift ... 5552
TeethPeriod ... 5553
TeethShift ... 5554
LipsPeriod ... 5555
LipsShift ... 5556
MaMethod ... 5557
Applied ... 5558
Create ... 5559
Jaw ... 5560
Teeth ... 5561
Lips ... 5562
Type ... 5563
CiAO ... 5564
Create ... 5566
Main ... 5567
Type ... 5568
CiFractals ... 5569
Create ... 5571
Upper ... 5572
Lower ... 5573
Type ... 5574
CiGator ... 5575
JawPeriod ... 5577
JawShift ... 5578
TeethPeriod ... 5579
TeethShift ... 5580
LipsPeriod ... 5581
LipsShift ... 5582
MaMethod ... 5583
Applied ... 5584
Create ... 5585
Upper ... 5586
Lower ... 5587
Type ... 5588
CiBWMFI ... 5589
Applied ... 5591
Create ... 5592
Main ... 5593
Type ... 5594
カスタム指標 ... 5595
NumBuffers ... 5596
NumParams ... 5597
ParamType ... 5598
ParamLong ... 5599
ParamDouble ... 5600
ParamString ... 5601
Type ... 5602
トレードクラス ... 5603
CAccountInfo ... 5604
Login ... 5606
TradeMode ... 5607
TradeModeDescription ... 5608
Leverage ... 5609

© 2000-2025, MetaQuotes Ltd.

StopoutMode 5610
StopoutModeDescription 5611
MarginMode 5612
MarginModeDescription 5613
TradeAllowed 5614
TradeExpert 5615
LimitOrders 5616
Balance 5617
Credit 5618
Profit 5619
Equity 5620
Margin 5621
FreeMargin 5622
MarginLevel 5623
MarginCall 5624
MarginStopOut 5625
Name 5626
Server 5627
Currency 5628
Company 5629
InfoInteger 5630
InfoDouble 5631
InfoString 5632
OrderProfitCheck 5633
MarginCheck 5634
FreeMarginCheck 5635
MaxLotCheck 5636
CSymbolInfo 5637
Refresh 5641
RefreshRates 5642
Name 5643
Select 5644
IsSynchronized 5645
Volume 5646
VolumeHigh 5647
VolumeLow 5648
Time 5649
Spread 5650
SpreadFloat 5651
TicksBookDepth 5652
StopsLevel 5653
FreezeLevel 5654
Bid 5655
BidHigh 5656
BidLow 5657
Ask 5658
AskHigh 5659
AskLow 5660
Last 5661
LastHigh 5662
LastLow 5663
TradeCalcMode 5664
TradeCalcModeDescription 5665
TradeMode 5666
TradeModeDescription 5667
TradeExecution 5668
TradeExecutionDescription 5669
SwapMode 5670

© 2000-2025, MetaQuotes Ltd.

SwapModeDescription ... 5671
SwapRollover3days ... 5672
SwapRollover3daysDescription ... 5673
MarginInitial ... 5674
MarginMaintenance ... 5675
MarginLong ... 5676
MarginShort ... 5677
MarginLimit ... 5678
MarginStop ... 5679
MarginStopLimit ... 5680
TradeTimeFlags ... 5681
TradeFillFlags ... 5682
Digits ... 5683
Point ... 5684
TickValue ... 5685
TickValueProfit ... 5686
TickValueLoss ... 5687
TickSize ... 5688
ContractSize ... 5689
LotsMin ... 5690
LotsMax ... 5691
LotsStep ... 5692
LotsLimit ... 5693
SwapLong ... 5694
SwapShort ... 5695
CurrencyBase ... 5696
CurrencyProfit ... 5697
CurrencyMargin ... 5698
Bank ... 5699
Description ... 5700
Path ... 5701
SessionDeals ... 5702
SessionBuyOrders ... 5703
SessionSellOrders ... 5704
SessionTurnover ... 5705
SessionInterest ... 5706
SessionBuyOrdersVolume ... 5707
SessionSellOrdersVolume ... 5708
SessionOpen ... 5709
SessionClose ... 5710
SessionAW ... 5711
SessionPriceSettlement ... 5712
SessionPriceLimitMin ... 5713
SessionPriceLimitMax ... 5714
InfoInteger ... 5715
InfoDouble ... 5716
InfoString ... 5717
NormalizePrice ... 5718
COrderInfo ... 5719
Ticket ... 5721
TimeSetup ... 5722
TimeSetupMsc ... 5723
OrderType ... 5724
TypeDescription ... 5725
State ... 5726
StateDescription ... 5727
TimeExpiration ... 5728
TimeDone ... 5729

© 2000-2025, MetaQuotes Ltd.

TimeDoneMsc 5730
TypeFilling 5731
TypeFillingDescription 5732
TypeTime 5733
TypeTimeDescription 5734
Magic 5735
PositionId 5736
VolumeInitial 5737
VolumeCurrent 5738
PriceOpen 5739
StopLoss 5740
TakeProfit 5741
PriceCurrent 5742
PriceStopLimit 5743
Symbol 5744
Comment 5745
InfoInteger 5746
InfoDouble 5747
InfoString 5748
StoreState 5749
CheckState 5750
Select 5751
SelectByIndex 5752
CHistoryOrderInfo 5753
TimeSetup 5755
TimeSetupMsc 5756
OrderType 5757
TypeDescription 5758
State 5759
StateDescription 5760
TimeExpiration 5761
TimeDone 5762
TimeDoneMsc 5763
TypeFilling 5764
TypeFillingDescription 5765
TypeTime 5766
TypeTimeDescription 5767
Magic 5768
PositionId 5769
VolumeInitial 5770
VolumeCurrent 5771
PriceOpen 5772
StopLoss 5773
TakeProfit 5774
PriceCurrent 5775
PriceStopLimit 5776
Symbol 5777
Comment 5778
InfoInteger 5779
InfoDouble 5780
InfoString 5781
Ticket 5782
SelectByIndex 5783
CPositionInfo 5784
Time 5786
TimeMsc 5787
TimeUpdate 5788
TimeUpdateMsc 5789

© 2000-2025, MetaQuotes Ltd.

PositionType ... 5790
TypeDescription ... 5791
Magic ... 5792
Identifier ... 5793
Volume ... 5794
PriceOpen ... 5795
StopLoss ... 5796
TakeProfit ... 5797
PriceCurrent ... 5798
Commission ... 5799
Swap ... 5800
Profit ... 5801
Symbol ... 5802
Comment ... 5803
InfoInteger ... 5804
InfoDouble ... 5805
InfoString ... 5806
Select ... 5807
SelectByIndex ... 5808
SelectByMagic ... 5809
SelectByTicket ... 5810
StoreState ... 5811
CheckState ... 5812
CDealInfo ... 5813
Order ... 5815
Time ... 5816
TimeMsc ... 5817
DealType ... 5818
TypeDescription ... 5819
Entry ... 5820
EntryDescription ... 5821
Magic ... 5822
PositionId ... 5823
Volume ... 5824
Price ... 5825
Commision ... 5826
Swap ... 5827
Profit ... 5828
Symbol ... 5829
Comment ... 5830
InfoInteger ... 5831
InfoDouble ... 5832
InfoString ... 5833
Ticket ... 5834
SelectByIndex ... 5835
CTrade ... 5836
LogLevel ... 5840
SetExpertMagicNumber ... 5841
SetDeviationInPoints ... 5842
SetTypeFilling ... 5843
SetTypeFillingBySymbol ... 5844
SetAsyncMode ... 5845
SetMarginMode ... 5846
OrderOpen ... 5847
OrderModify ... 5849
OrderDelete ... 5850
PositionOpen ... 5851
PositionModify ... 5852

© 2000-2025, MetaQuotes Ltd.

PositionClose ... 5853
PositionClosePartial ... 5854
PositionCloseBy ... 5856
Buy ... 5857
Sell ... 5858
BuyLimit ... 5859
BuyStop ... 5860
SellLimit ... 5861
SellStop ... 5862
Request ... 5863
RequestAction ... 5864
RequestActionDescription ... 5865
RequestMagic ... 5866
RequestOrder ... 5867
RequestSymbol ... 5868
RequestVolume ... 5869
RequestPrice ... 5870
RequestStopLimit ... 5871
RequestSL ... 5872
RequestTP ... 5873
RequestDeviation ... 5874
RequestType ... 5875
RequestTypeDescription ... 5876
RequestTypeFilling ... 5877
RequestTypeFillingDescription ... 5878
RequestTypeTime ... 5879
RequestTypeTimeDescription ... 5880
RequestExpiration ... 5881
RequestComment ... 5882
RequestPosition ... 5883
RequestPositionBy ... 5884
Result ... 5885
ResultRetcode ... 5886
ResultRetcodeDescription ... 5887
ResultDeal ... 5888
ResultOrder ... 5889
ResultVolume ... 5890
ResultPrice ... 5891
ResultBid ... 5892
ResultAsk ... 5893
ResultComment ... 5894
CheckResult ... 5895
CheckResultRetcode ... 5896
CheckResultRetcodeDescription ... 5897
CheckResultBalance ... 5898
CheckResultEquity ... 5899
CheckResultProfit ... 5900
CheckResultMargin ... 5901
CheckResultMarginFree ... 5902
CheckResultMarginLevel ... 5903
CheckResultComment ... 5904
PrintRequest ... 5905
PrintResult ... 5906
FormatRequest ... 5907
FormatRequestResult ... 5908
CTerminalInfo ... 5909
Build ... 5911
IsConnected ... 5912

© 2000-2025, MetaQuotes Ltd.

IsDLLsAllowed 5913
IsTradeAllowed 5914
IsEmailEnabled 5915
IsFtpEnabled 5916
MaxBars 5917
CodePage 5918
CPUCores 5919
MemoryPhysical 5920
MemoryTotal 5921
MemoryAvailable 5922
MemoryUsed 5923
IsX64 5924
OpenCLSupport 5925
DiskSpace 5926
Language 5927
Name 5928
Company 5929
Path 5930
DataPath 5931
CommonDataPath 5932
InfoInteger 5933
InfoString 5934
ストラテジーモジュール 5935
エキスパートアドバイザーの基本クラス 5937
CExpertBase 5938
InitPhase 5940
TrendType 5941
UsedSeries 5942
EveryTick 5943
Open 5944
High 5945
Low 5946
Close 5947
Spread 5948
Time 5949
TickVolume 5950
RealVolume 5951
Init 5952
Symbol 5953
Period 5954
Magic 5955
ValidationSettings 5956
SetPriceSeries 5957
SetOtherSeries 5958
InitIndicators 5959
InitOpen 5960
InitHigh 5961
InitLow 5962
InitClose 5963
InitSpread 5964
InitTime 5965
InitTickVolume 5966
InitRealVolume 5967
PriceLevelUnit 5968
StartIndex 5969
CompareMagic 5970
CExpert 5971
Init 5976

© 2000-2025, MetaQuotes Ltd.

Magic 5977
InitSignal 5978
InitTrailing 5979
InitMoney 5980
InitTrade 5981
Deinit 5982
OnTickProcess 5983
OnTradeProcess 5984
OnTimerProcess 5985
OnChartEventProcess 5986
OnBookEventProcess 5987
MaxOrders 5988
Signal 5989
ValidationSettings 5990
InitIndicators 5991
OnTick 5992
OnTrade 5993
OnTimer 5994
OnChartEvent 5995
OnBookEvent 5996
InitParameters 5997
DeinitTrade 5998
DeinitSignal 5999
DeinitTrailing 6000
DeinitMoney 6001
DeinitIndicators 6002
Refresh 6003
Processing 6004
SelectPosition 6006
CheckOpen 6007
CheckOpenLong 6008
CheckOpenShort 6009
OpenLong 6010
OpenShort 6011
CheckReverse 6012
CheckReverseLong 6013
CheckReverseShort 6014
ReverseLong 6015
ReverseShort 6017
CheckClose 6019
CheckCloseLong 6020
CheckCloseShort 6021
CloseAll 6022
Close 6023
CloseLong 6024
CloseShort 6025
CheckTrailingStop 6026
CheckTrailingStopLong 6027
CheckTrailingStopShort 6028
TrailingStopLong 6029
TrailingStopShort 6030
CheckTrailingOrderLong 6031
CheckTrailingOrderShort 6032
TrailingOrderLong 6033
TrailingOrderShort 6034
CheckDeleteOrderLong 6035
CheckDeleteOrderShort 6036
DeleteOrders 6037

© 2000-2025, MetaQuotes Ltd.

DeleteOrder ... 6038
DeleteOrderLong ... 6039
DeleteOrderShort ... 6040
LotOpenLong ... 6041
LotOpenShort ... 6042
LotReverse ... 6043
PrepareHistoryDate ... 6044
HistoryPoint ... 6045
CheckTradeState ... 6046
WaitEvent ... 6047
NoWaitEvent ... 6048
TradeEventPositionStopTake ... 6049
TradeEventOrderTriggered ... 6050
TradeEventPositionOpened ... 6051
TradeEventPositionVolumeChanged ... 6052
TradeEventPositionModified ... 6053
TradeEventPositionClosed ... 6054
TradeEventOrderPlaced ... 6055
TradeEventOrderModified ... 6056
TradeEventOrderDeleted ... 6057
TradeEventNotIdentified ... 6058
TimeframeAdd ... 6059
TimeframesFlags ... 6060
CExpertSignal ... 6061
BasePrice ... 6064
UsedSeries ... 6065
Weight ... 6066
PatternsUsage ... 6067
General ... 6068
Ignore ... 6069
Invert ... 6070
ThresholdOpen ... 6071
ThresholdClose ... 6072
PriceLevel ... 6073
StopLevel ... 6074
TakeLevel ... 6075
Expiration ... 6076
Magic ... 6077
ValidationSettings ... 6078
InitIndicators ... 6079
AddFilter ... 6080
CheckOpenLong ... 6081
CheckOpenShort ... 6082
OpenLongParams ... 6083
OpenShortParams ... 6084
CheckCloseLong ... 6085
CheckCloseShort ... 6086
CloseLongParams ... 6087
CloseShortParams ... 6088
CheckReverseLong ... 6089
CheckReverseShort ... 6090
CheckTrailingOrderLong ... 6091
CheckTrailingOrderShort ... 6092
LongCondition ... 6093
ShortCondition ... 6094
Direction ... 6095
CExpertTrailing ... 6096
CheckTrailingStopLong ... 6097

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort ... 6098
CExpertMoney ... 6099
Percent ... 6100
ValidationSettings ... 6101
CheckOpenLong ... 6102
CheckOpenShort ... 6103
CheckReverse ... 6104
CheckClose ... 6105
売買シグナルのモジュール ... 6106
AC オシレーター（Accelerator Oscillator）指標のシグナル ... 6109
適応型移動平均（Adaptive Moving Average）指標のシグナル ... 6112
オーサムオシレーター（Awesome Oscillator）指標のシグナル ... 6116
ベアパワー（Bears Power）オシレーター指標のシグナル ... 6120
ブルパワー（Bulls Power）オシレーター指標のシグナル ... 6122
CCI(Commodity Channel Index、コモディティチャンネルインデックス)指標のシグナル ... 6124
デマーカー（DeMarker）オシレーター指標のシグナル ... 6128
2 重指数移動平均（Double Exponential Moving Average）指標のシグナル ... 6132
エンベロープ（Envelopes）指標のシグナル ... 6136
フラクタル適応型移動平均（Fractal Adaptive Moving Average）指標のシグナル ... 6139
日中時間フィルタのシグナル ... 6143
MACD オシレーターのシグナル ... 6144
移動平均（Moving Average）指標のシグナル ... 6150
パラボリック SAR（Parabolic SAR） 指標のシグナル ... 6154
相対力指数（Relative Strength Index）オシレーターのシグナル ... 6156
相対力指数（Relative Vigor Index）オシレーターのシグナル ... 6162
ストキャスティックス（Stochastic）のシグナル ... 6164
3 重指数平均（Triple Exponential Average）オシレーターのシグナル ... 6169
3 重指数移動平均（Triple Exponential Moving Average）指標のシグナル ... 6173
ウィリアムパーセントレンジ（Williams Percent Range）オシレーターのシグナル ... 6177
トレール注文のクラス ... 6180
CTrailingFixedPips ... 6181
StopLevel ... 6183
ProfitLevel ... 6184
ValidationSettings ... 6185
CheckTrailingStopLong ... 6186
CheckTrailingStopShort ... 6187
CTrailingMA ... 6188
Period ... 6189
Shift ... 6190
Method ... 6191
Applied ... 6192
InitIndicators ... 6193
ValidationSettings ... 6194
CheckTrailingStopLong ... 6195
CheckTrailingStopShort ... 6196
CTrailingNone ... 6197
CheckTrailingStopLong ... 6198
CheckTrailingStopShort ... 6199
CTrailingPSAR ... 6200
Step ... 6201
Maximum ... 6202
InitIndicators ... 6203
CheckTrailingStopLong ... 6204
CheckTrailingStopShort ... 6205
資金管理のクラス ... 6206
CMoneyFixedLot ... 6207

© 2000-2025, MetaQuotes Ltd.

Lots 6208
ValidationSettings 6209
CheckOpenLong 6210
CheckOpenShort 6211
CMoneyFixedMargin 6212
CheckOpenLong 6213
CheckOpenShort 6214
CMoneyFixedRisk 6215
CheckOpenLong 6216
CheckOpenShort 6217
CMoneyNone 6218
ValidationSettings 6219
CheckOpenLong 6220
CheckOpenShort 6221
CMoneySizeOptimized 6222
DecreaseFactor 6223
ValidationSettings 6224
CheckOpenLong 6225
CheckOpenShort 6226
パネルとダイアログ 6227
CRect 6229
Left 6230
Top 6231
Right 6232
Bottom 6233
Width 6234
Height 6235
SetBound 6236
Move 6237
Shift 6238
Contains 6239
Format 6240
CDateTime 6241
MonthName 6243
ShortMonthName 6244
DayName 6245
ShortDayName 6246
DaysInMonth 6247
DateTime 6248
Date 6249
Time 6250
Sec 6251
Min 6252
Hour 6253
Day 6254
Mon 6255
Year 6256
SecDec 6257
SecInc 6258
MinDec 6259
MinInc 6260
HourDec 6261
HourInc 6262
DayDec 6263
DayInc 6264
MonDec 6265
MonInc 6266
YearDec 6267

© 2000-2025, MetaQuotes Ltd.

YearInc 6268
CWnd 6269
Create 6273
Destroy 6274
OnEvent 6275
OnMouseEvent 6276
Name 6277
ControlsTotal 6278
Control 6279
ControlFind 6280
Rect 6281
Left 6282
Top 6283
Right 6284
Bottom 6285
Width 6286
Height 6287
Move 6288
Shift 6289
Resize 6290
Contains 6291
Alignment 6292
Align 6293
Id 6294
IsEnabled 6295
Enable 6296
Disable 6297
IsVisible 6298
Visible 6299
Show 6300
Hide 6301
IsActive 6302
Activate 6303
Deactivate 6304
StateFlags 6305
StateFlagsSet 6306
StateFlagsReset 6307
PropFlags 6308
PropFlagsSet 6309
PropFlagsReset 6310
MouseX 6311
MouseY 6312
MouseFlags 6313
MouseFocusKill 6314
OnCreate 6315
OnDestroy 6316
OnMove 6317
OnResize 6318
OnEnable 6319
OnDisable 6320
OnShow 6321
OnHide 6322
OnActivate 6323
OnDeactivate 6324
OnClick 6325
OnChange 6326
OnMouseDown 6327
OnMouseUp 6328

© 2000-2025, MetaQuotes Ltd.

OnDragStart ... 6329
OnDragProcess ... 6330
OnDragEnd ... 6331
DragObjectCreate ... 6332
DragObjectDestroy ... 6333
CWndObj ... 6334
OnEvent ... 6336
Text ... 6337
Color ... 6338
ColorBackground ... 6339
ColorBorder ... 6340
Font ... 6341
FontSize ... 6342
ZOrder ... 6343
OnObjectCreate ... 6344
OnObjectChange ... 6345
OnObjectDelete ... 6346
OnObjectDrag ... 6347
OnSetText ... 6348
OnSetColor ... 6349
OnSetColorBackground ... 6350
OnSetFont ... 6351
OnSetFontSize ... 6352
OnSetZOrder ... 6353
OnDestroy ... 6354
OnChange ... 6355
CWndContainer ... 6356
Destroy ... 6358
OnEvent ... 6359
OnMouseEvent ... 6360
ControlsTotal ... 6361
Control ... 6362
ControlFind ... 6363
Add ... 6364
Delete ... 6365
Move ... 6366
Shift ... 6367
Id ... 6368
Enable ... 6369
Disable ... 6370
Show ... 6371
Hide ... 6372
MouseFocusKill ... 6373
Save ... 6374
Load ... 6375
OnResize ... 6376
OnActivate ... 6377
OnDeactivate ... 6378
CLabel ... 6379
Create ... 6384
OnSetText ... 6385
OnSetColor ... 6386
OnSetFont ... 6387
OnSetFontSize ... 6388
OnCreate ... 6389
OnShow ... 6390
OnHide ... 6391
OnMove ... 6392

© 2000-2025, MetaQuotes Ltd.

CBmpButton ... 6393
Create ... 6400
Border ... 6401
BmpNames ... 6402
BmpOffName ... 6403
BmpOnName ... 6404
BmpPassiveName ... 6405
BmpActiveName ... 6406
Pressed ... 6407
Locking ... 6408
OnSetZOrder ... 6409
OnCreate ... 6410
OnShow ... 6411
OnHide ... 6412
OnMove ... 6413
OnChange ... 6414
OnActivate ... 6415
OnDeactivate ... 6416
OnMouseDown ... 6417
OnMouseUp ... 6418
CButton ... 6419
Create ... 6426
Pressed ... 6427
Locking ... 6428
OnSetText ... 6429
OnSetColor ... 6430
OnSetColorBackground ... 6431
OnSetColorBorder ... 6432
OnSetFont ... 6433
OnSetFontSize ... 6434
OnCreate ... 6435
OnShow ... 6436
OnHide ... 6437
OnMove ... 6438
OnResize ... 6439
OnMouseDown ... 6440
OnMouseUp ... 6441
CEdit ... 6442
Create ... 6447
ReadOnly ... 6448
TextAlign ... 6449
OnObjectEndEdit ... 6450
OnSetText ... 6451
OnSetColor ... 6452
OnSetColorBackground ... 6453
OnSetColorBorder ... 6454
OnSetFont ... 6455
OnSetFontSize ... 6456
OnSetZOrder ... 6457
OnCreate ... 6458
OnShow ... 6459
OnHide ... 6460
OnMove ... 6461
OnResize ... 6462
OnChange ... 6463
OnClick ... 6464
CPanel ... 6465
Create ... 6470

© 2000-2025, MetaQuotes Ltd.

BorderType 6471
OnSetText 6472
OnSetColorBackground 6473
OnSetColorBorder 6474
OnCreate 6475
OnShow 6476
OnHide 6477
OnMove 6478
OnResize 6479
OnChange 6480
CPicture 6481
Create 6486
Border 6487
BmpName 6488
OnCreate 6489
OnShow 6490
OnHide 6491
OnMove 6492
OnChange 6493
CScroll 6494
Create 6496
OnEvent 6497
MinPos 6498
MaxPos 6499
CurrPos 6500
CreateBack 6501
CreateInc 6502
CreateDec 6503
CreateThumb 6504
OnClickInc 6505
OnClickDec 6506
OnShow 6507
OnHide 6508
OnChangePos 6509
OnThumbDragStart 6510
OnThumbDragProcess 6511
OnThumbDragEnd 6512
CalcPos 6513
CScrollV 6514
CreateInc 6520
CreateDec 6521
CreateThumb 6522
OnResize 6523
OnChangePos 6524
OnThumbDragStart 6525
OnThumbDragProcess 6526
OnThumbDragEnd 6527
CalcPos 6528
CScrollH 6529
CreateInc 6535
CreateDec 6536
CreateThumb 6537
OnResize 6538
OnChangePos 6539
OnThumbDragStart 6540
OnThumbDragProcess 6541
OnThumbDragEnd 6542
CalcPos 6543

© 2000-2025, MetaQuotes Ltd.

CWndClient 6544
Create 6546
OnEvent 6547
ColorBackground 6548
ColorBorder 6549
BorderType 6550
VScrolled 6551
HScrolled 6552
CreateBack 6553
CreateScrollV 6554
CreateScrollH 6555
OnResize 6556
OnVScrollShow 6557
OnVScrollHide 6558
OnHScrollShow 6559
OnHScrollHide 6560
OnScrollLineDown 6561
OnScrollLineUp 6562
OnScrollLineLeft 6563
OnScrollLineRight 6564
Rebound 6565
CListView 6566
Create 6572
OnEvent 6573
TotalView 6574
AddItem 6575
Select 6576
SelectByText 6577
SelectByValue 6578
Value 6579
CreateRow 6580
OnResize 6581
OnVScrollShow 6582
OnVScrollHide 6583
OnScrollLineDown 6584
OnScrollLineUp 6585
OnItemClick 6586
Redraw 6587
RowState 6588
CheckView 6589
CComboBox 6590
Create 6596
OnEvent 6597
AddItem 6598
ListViewItems 6599
Select 6600
SelectByText 6601
SelectByValue 6602
Value 6603
CreateEdit 6604
CreateButton 6605
CreateList 6606
OnClickEdit 6607
OnClickButton 6608
OnChangeList 6609
ListShow 6610
ListHide 6611
CCheckBox 6612

© 2000-2025, MetaQuotes Ltd.

Create ... 6618
OnEvent ... 6619
Text ... 6620
Color ... 6621
Checked ... 6622
Value ... 6623
CreateButton ... 6624
CreateLabel ... 6625
OnClickButton ... 6626
OnClickLabel ... 6627
CCheckGroup ... 6628
Create ... 6634
OnEvent ... 6635
AddItem ... 6636
Value ... 6637
CreateButton ... 6638
OnVScrollShow ... 6639
OnVScrollHide ... 6640
OnScrollLineDown ... 6641
OnScrollLineUp ... 6642
OnChangeItem ... 6643
Redraw ... 6644
RowState ... 6645
CRadioButton ... 6646
Create ... 6648
OnEvent ... 6649
Text ... 6650
Color ... 6651
State ... 6652
CreateButton ... 6653
CreateLabel ... 6654
OnClickButton ... 6655
OnClickLabel ... 6656
CRadioGroup ... 6657
Create ... 6663
OnEvent ... 6664
AddItem ... 6665
Value ... 6666
CreateButton ... 6667
OnVScrollShow ... 6668
OnVScrollHide ... 6669
OnScrollLineDown ... 6670
OnScrollLineUp ... 6671
OnChangeItem ... 6672
Redraw ... 6673
RowState ... 6674
Select ... 6675
CSpinEdit ... 6676
Create ... 6681
OnEvent ... 6682
MinValue ... 6683
MaxValue ... 6684
Value ... 6685
CreateEdit ... 6686
CreateInc ... 6687
CreateDec ... 6688
OnClickInc ... 6689
OnClickDec ... 6690

© 2000-2025, MetaQuotes Ltd.

OnChangeValue ... 6691
CDialog ... 6692
Create ... 6694
OnEvent ... 6695
Caption ... 6696
Add ... 6697
CreateWhiteBorder ... 6698
CreateBackground ... 6699
CreateCaption ... 6700
CreateButtonClose ... 6701
CreateClientArea ... 6702
OnClickCaption ... 6703
OnClickButtonClose ... 6704
ClientAreaVisible ... 6705
ClientAreaLeft ... 6706
ClientAreaTop ... 6707
ClientAreaRight ... 6708
ClientAreaBottom ... 6709
ClientAreaWidth ... 6710
ClientAreaHeight ... 6711
OnDialogDragStart ... 6712
OnDialogDragProcess ... 6713
OnDialogDragEnd ... 6714
CAppDialog ... 6715
Create ... 6718
Destroy ... 6719
OnEvent ... 6720
Run ... 6721
ChartEvent ... 6722
Minimized ... 6723
IniFileSave ... 6724
IniFileLoad ... 6725
IniFileName ... 6726
IniFileExt ... 6727
CreateCommon ... 6728
CreateExpert ... 6729
CreateIndicator ... 6730
CreateButtonMinMax ... 6731
OnClickButtonClose ... 6732
OnClickButtonMinMax ... 6733
OnAnotherApplicationClose ... 6734
Rebound ... 6735
Minimize ... 6736
Maximize ... 6737
CreateInstanceId ... 6738
ProgramName ... 6739
SubwinOff ... 6740
36 MQL4 からの移行 ... 6741
37 MQL5関数の表 ... 6744
38 MQL5定数の表 ... 6772

© 2000-2025, MetaQuotes Ltd.

MQL5 레퍼런스

MQL5(MetaQuotes Language 5)는 금융 거래를 자동화하는 기술 지표, 트레이딩 로봇 및 유틸리티 애플리케이션을 개발하기 위해 고안된 고급 언어입니다. MQL5 는 MetaQuotes 에서 거래 플랫폼을 위해 개발했습니다. 언어 구문은 프로그래머가 개체 지향 프로그래밍(OOP) 스타일로 애플리케이션을 개발할 수 있도록 C++와 매우 유사합니다.

트레이딩 플랫폼 패키지에는 MQL5 언어 외에도 템플릿, 정보, 디버깅, 프로파일링 및 자동완성 툴과 같은 고급 코드 쓰기 툴을 포함하는 MetaEditor IDE가 포함되며, 또한 파일 버전 관리를 지원하는 내장 MQL5 스토리지도 포함되어 있습니다.

언어 지원은 MQL5 Algotrading 커뮤니티 웹사이트에서 이용할 수 있으며, 이 웹사이트에는 방대한 무료 코드베이스와 다양한 아티클이 포함되어 있습니다. 이 아티클에서 뉴럴 네트워크, 통계 및 분석, 고빈도 트레이딩, 차익 거래, 트레이딩 전략의 테스트 및 최적화, 트레이딩 자동 로봇 사용 등을 포함한 현대 거래의 모든 측면을 다룹니다.

트레이더와 MQL5 프로그램 개발자는 포럼에서 소통하고 프리랜스 서비스를 이용하여 애플리케이션을 주문 및 개발할 수 있으며, 자동화된 트레이딩 애플리케이션 마켓에서 보호된 프로그램을 구매 및 판매할 수 있습니다.

MQL5 언어는 전문화된 트레이딩 기능과 사전 정의된 이벤트 처리기를 제공하여 프로그래머가 특정 거래 규칙에 따라 거래 프로세스를 자동으로 제어하는 EA(엑스퍼트 어드바이저)를 개발할 수 있도록 지원합니다. EA외에도 MQL5를 통해 맞춤형 기술 지표, 스크립트 및 라이브러리를 개발할 수 있습니다.

이 MQL5 언어 참조에는 범주별로 구분된 함수, 연산, 예약된 단어 및 기타 언어 구성이 포함되어 있습니다. 또한 트레이딩 전략, 제어판, 사용자 지정 그래픽 및 액세스 활성화에 사용되는 표준 라이브러 클래스에 대한 설명을 제공합니다.

또한 코드베이스에는 다양한 수학 문제를 해결하는 데 사용할 수 있는 ALGLIB 수치 분석 라이브러리가 포함되어 있습니다.

알고 트레이딩 북

새로운 것을 배우기 시작하는 것은 언제나 어려운 일입니다. 초보자를 돕기 위해 우리는 트레이딩 로봇을 생성하고 알고리즘 거래용 애플리케이션을 마스터하려는 모든 사람들을 위해 이에 대해 전반적으로 다루는 MQL5 프로그래밍에 대한 두 권의 책을 출시했습니다.

이 책들은 학습 과정을 훨씬 더 쉽게 만들기 위해 체계적이고 구조화된 자료를 제시합니다. 트레이딩 로봇과 애플리케이션의 단계별 생성을 설명하는 자세한 코드 예제를 통해 알고리즘 트레이딩의 미묘한 차이를 더 깊이 이해할 수 있습니다. 이 책에는 여러분이 습득한 지식을 강화하고 실제 거래 환경에서 프로그래밍 기술을 개발하는 데 도움이 되는 수많은 실제 연습 예제들이 포함되어 있습니다.

"트레이더를 위한 MQL5 프로그래밍"은 모든 수준의 프로그래머에게 적합한 MQL5에 대한 가장 완전한 튜토리얼입니다. 초보자는 기본 사항에 대해 배우게 됩니다: 이 책에서는 개발 도구와 기본 프로그래밍 개념을 소개합니다. 이 자료를 바탕으로 MetaTrader 5 거래 플랫폼에서 첫 번째 애플리케이션을 생성하고 컴파일 및 실행하게 됩니다. 다른 프로그래밍 언어에 대한 경험이 있는 사용자는 MQL5에서 트레이딩 로봇 및 분석 애플리케이션을 생성하는 애플리케이션 섹션부터 즉시 진행할 수 있습니다.

"MQL5를 사용한 알고리즘 거래를 위한 신경망"는 MetaTrader 5 플랫폼용 로봇 거래에서 머신 러닝을 사용하는 방법에 대한 가이드입니다. 여러분은 신경망의 기초와 알고리즘 거래에서의 적용에 대해 알게될

© 2000-2025, MetaQuotes Ltd.

것입니다. 그렇게 여러분 스스로 발전하면서 자신만의 AI 솔루션을 구축하고 훈련하면서 점차적으로 새로운 기능을 추가하게될 것입니다. 여러분은 MQL5에 대해 알아 가는것 이외에도 Python 및 OpenCL 프로그래밍 기술을 습득하고 간결하고 효율적인 코드로 복잡한 수학적 문제를 해결할 수 있는 행렬 및 벡터에 관한 통합적인 메서드를 배우게 될 것입니다.

트레이딩 애플리케이션 개발에 관한 기사

MQL5 기사들은 광범위한 실제의 알고리즘 트레이딩 작업을 다루면서 이들 프로그래밍 언어가 가진 전체적인 잠재력을 탐색할 수 있는 훌륭한 리소스입니다. 쉽게 탐색할 수 있도록 모든 기사는 예제, 전문가 조언, <와 같은 섹션으로 분류되어 있습니다. li5>머신 러닝 등. 매달 MQL5 Algotrading 커뮤니티 웹사이트에는 여러 트레이더들이 다른 트레이더들을 위해 작성한 수십 개의 새로운 기사가 게시됩니다. 여러분이 최신 알고리즘 거래를 마스터하고자 한다면 이 기사를 읽고 의견을 나눠보세요. 저희가 초보자들이 MQL5에 빠르게 몰입할 수 있도록 16개의 추천 기사 목록을 정리했습니다.

MQL5 애플리케이션 유형

MQL5 프로그램은 다음과 같은 트레이딩 자동화 작업에 따라 4가지 전문 유형으로 나뉩니다:

- **엑스퍼트 어드바이저**는 차트에 연결된 자동 트레이딩 시스템입니다. 엑스퍼트 어드바이저에는 적절한 트레이딩 전략 요소 실행을 활성화 하는 사전 정의된 이벤트를 관리하는 이벤트 처리기가 포함되어 있습니다. 예를 들어 프로그램 초기화 및 초기화 해제 이벤트, 새 틱, 타이머 이벤트, 시장 수준 변경, 차트 및 사용자 지정 이벤트 등이 있습니다.
  엑스퍼트 어드바이저는 구현된 규칙에 따라 트레이딩 시그널을 계산하는 것 외에도 거래를 자동으로 실행하여 거래 서버로 직접 전송할 수도 있습니다. 엑스퍼트 어드바이저는 *<Terminal_Directory>\MQL5\Experts*에 저장됩니다.
- **맞춤형 지표**는 트레이딩 플랫폼에 통합된 표준 지표 외에 사용자가 개발한 기술 지표입니다. 표준 지표뿐만 아니라 맞춤형 지표도 자동으로 거래할 수 없고 분석 기능만 구현합니다. 맞춤형 지표는 계산을 위해 다른 지표 값을 활용할 수 있으며, 엑스퍼트 어드바이저에서 호출할 수 있습니다.
  맞춤형 지표는 *<Terminal_Directory>\MQL5\Indicators*에 저장됩니다.
- **스크립트**는 작업이 단일 실행을 위한 프로그램입니다. 엑스퍼트 어드바이저와 달리 스크리트는 트리거를 제외하고 이벤트를 처리하지 않습니다. 스크립트 코드에는 OnStart 처리기 기능이 포함되어야 합니다.
  스크립트는 *<Terminal_DIrectory>\MQL5\Scripts에 저장됩니다.*
- **서비스**는 지표와 달리 작동하기 위해 차트에 바인딩할 필요가 없는 프로그램입니다. 스크립트와 마찬가지로 서비스도 트리거를 제외하고 이벤트를 처리하지 않습니다. 서비스를 시작하려면 고트에 OnStart 처리기 기능이 포함되어 있어야 합니다. 서비스에서는 시작을 제외한 다른 이벤트를 허용하지 않지만 EventChartCustom를 사용하여 사용자 지정 이벤트를 차트에 보낼 수 있습니다. 서비스는 <terminal_directory>₩MQL5₩Services에 저장됩니다.
- **라이브러리** 는 사용자 지정 기능의 집합입니다. 라이브러리는 일반적으로 사용되는 사용자 지정 프로그램 알고리즘을 저장하고 배포하기 위한 것입니다.
  라이브러리는 *<Terminal_Directory>\MQL5\Libraries*에 저장됩니다.
- **포함 파일**은 가장 자주 사용되는 사용자 지정 프로그램 블록의 원본 텍스트 입니다. 이러한 파일은 컴파일 단계에서 엑스퍼트 어드바이저, 스크립트, 맞춤형 지표 및 라이브러리의 원복 텍스트에 포함될 수 있습니다. 라이브러리 기능 호출 시 추가적인 부담이 발생하기 때문에 포함 파일을 사용하는 것이 라이브러리를 사용하는 것보다 좋습니다.
  포함 파일은 원래 파일이 있는 디렉토리에 저장할 수 있습니다. 이 경우 큰 따옴표가 있는 #include 지

© 2000-2025, MetaQuotes Ltd.

시어가 사용됩니다. 또 다른 옵션은 포함 파일을 <*Terminal_Directory*>*MQL5**Include*에 저장하는 것입니다. 이 경우 꺾쇠괄호가 있는 #include를 사용해야 합니다.

© 2000-2025, MetaQuotes Ltd.

언어 기본사항

MQL5(MetaQuotes Language 5)는 다양한 금융 시장 분석을 위한 맞춤형 기술 지표인 자동 거래 전략을 작성하기 위한 객체 지향 고급 프로그래밍 언어입니다. 이를 통해 실시간으로 작동하도록 설계된 다양한 전문가 시스템을 작성할 수 있을 뿐만 아니라 거래 결정을 내리는 데 도움이 되는 자체 그래픽 도구를 만들 수 있습니다.

MQL5는 인기있는 프로그래밍 언어 C++의 개념을 기반으로 합니다. MQL4와 비교했을 때, 새로운 언어는 이제 열거, 구조, 클래스 및 이벤트 핸들링을 가집니다. 임베디드 메인 유형의 수를 늘림으로써, MQL5의 실행 가능한 프로그램들과 dll을 통한 다른 애플리케이션과 상호작용이 최대한 쉬워집니다. MQL5 구문은 C++의 구문과 유사하며, 이를 통해 현대 프로그래밍 언어에서 이 프로그램을 쉽게 번역할 수 있습니다.

MQL5 언어를 공부하는 데 도움이 되도록 모든 항목은 다음 섹션으로 그룹화 됩니다:

- 구문
- 데이터 유형
- 연산 및 표현식
- 연산자
- 함수
- 변수
- 전처리장치
- 객체 지향 프로그래밍
- 네임스페이스

© 2000-2025, MetaQuotes Ltd.

문법

구문과 관련하여, 거래 전략 프로그래밍을 위한 MQL5 언어는 일부 기능을 제외하고 C++ 프로그래밍 언어와 매우 유사합니다.

- 주소 산술 없음;
- goto 연산자 없음;
- 익명 열거값 선언 불가;
- 다중 상속 없음.

더 보기

열거값, 구조 및 클래스, 상속

© 2000-2025, MetaQuotes Ltd.

주석

멀티라인 주석은 /* 기호로 시작하여 */ 로 끝납니다. 이러한 종류의 주석은 중첩될 수 없습니다. 한줄 주석은 // 기호로 시작하고 줄넘김 문자로 끝나며 멀티라인 주석에 중첩될 수 있습니다. 주석은 공백이 허용되는 모든 곳에서 허용되며, 코멘트에 공백이 있을 수 있습니다.

예제:

```
//--- 한줄 주석
/*  멀티-
   라인          // 중첩된 한줄 주석
   주석
*/
```

© 2000-2025, MetaQuotes Ltd.

식별자

식별자는 변수 및 함수의 이름으로 사용됩니다. 식별자의 길이는 63자를 초과할 수 없습니다.

식별자에 쓸 수 있는 문자: 그림 0-9, 라틴 대/소문자 a-z 및 A-Z, 다른 문자로 인식되는 밑줄 문자(_). 첫 번째 문자는 숫자일 수 없습니다.

식별자는 지정된 단어와 같아서는 안됩니다.

예제:

```
NAME1 name1 Total_5 Paper
```

더 보기

변수, 함수

© 2000-2025, MetaQuotes Ltd.

지정 단어

다음 식별자는 지정된 단어로 기록되며 각 식별자는 특정 작업에 해당하며 다른 의미로는 사용할 수 없습니다.

데이터 타입

bool	float	uint
char	int	ulong
class	long	union
색상	short	ushort
datetime	string	void
double	struct	
enum	uchar	

Access Specificators

const	private	virtual
delete	protected	
override	public	

Memory Classes

extern	input	static

연산자

break	dynamic_cast	operator
case	else	pack
continue	for	return
default	if	sizeof
delete	new	switch
do	offsetof	while

Other

this	#define	#import
true	#ifdef	#include

© 2000-2025, MetaQuotes Ltd.

this	#define	#import
false	#ifndef	#property
template	#else	group
typename	#endif	namespace

© 2000–2025, MetaQuotes Ltd.

데이터 타입

모든 프로그램은 데이터로 작동합니다. 데이터는 용도에 따라 유형이 다를 수 있습니다. 예를 들어, 배열 구성 요소에 액세스하는 데 정수 데이터가 사용됩니다. 가격 데이터는 부동 소수점이 있는 2배 정밀도의 데이터에 속합니다. 이는 가격 데이터에 대한 특별한 데이터 유형이 MQL5에 제공되지 않는다는 사실과 관련이 있습니다.

다른 유형의 데이터는 다른 속도로 처리됩니다. 정수 데이터는 가장 빠른 속도로 처리됩니다. double 정밀 데이터를 처리하려면 특수 보조 프로세서가 사용됩니다. 그러나 부동 소수점으로 데이터를 내부적으로 표현하는 작업이 복잡하기 때문에 정수보다 느리게 처리됩니다.

동적 컴퓨터 메모리 할당/재할당으로 인해 문자열 데이터는 최대로 처리됩니다.

기본 데이터 유형은 다음과 같습니다:

- integers (char, short, int, long, uchar, ushort, uint, ulong);
- logical (bool);
- literals (ushort);
- strings (string);
- floating-point numbers (double, float);
- color (color);
- date and time (datetime);
- enumerations (enum).

복합 데이터 유형은 다음과 같습니다:

- structures;
- classes.

OOP 복합 데이터 유형을 추상 데이터 유형이라고 합니다.

color 및 *datetime* 유형은 Expert Advisor 또는 사용자 지정 지표 속성 테이블(입력 탭)에서 외부에서 정의된 매개변수를 쉽게 시각화하고 입력할 수 있는 경우에만 적합합니다. *color* 및 *datetime* 유형의 데이터는 정수로 표시됩니다. 정수 유형과 부동 소수점 유형을 산술(숫자) 유형이라고 합니다.

캐스팅을 명시적으로 지정하지 않는 한, 암시적 타입 캐스팅만이 표현식에 사용됩니다.

더 보기

타입캐스팅

© 2000-2025, MetaQuotes Ltd.

정수 타입

MQL5에서 정수는 11가지 유형으로 표시됩니다. 프로그램 논리에 필요한 경우 일부 유형은 다른 유형과 함께 사용할 수 있지만, 이 경우 타입캐스팅 규칙을 기억해야 합니다.

아래 표에는 각 유형의 특성이 나열되어 있습니다. 게다가 마지막 열에는 각 유형에 해당하는 C++의 유형이 있습니다.

유형	크기(바이트)	최소값	최소값	C++ 아날로그
char	1	-128	127	char
uchar	1	0	255	unsigned char, BYTE
bool	1	0(false)	1(true)	bool
short	2	-32 768	32 767	short, wchar_t
ushort	2	0	65 535	unsigned short, WORD
int	4	- 2 147 483 648	2 147 483 647	int
uint	4	0	4 294 967 295	unsigned int, DWORD
색상	4	-1	16 777 215	int, COLORREF
long	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807	__int64
ulong	8	0	18 446 744 073 709 551 615	unsigned __int64
datetime	8	0 (1970.01.01 0:00:00)	32 535 244 799 (3000.12.31 23:59:59)	__time64_t

소스 TTS복사하기번역하기자동완성정수 유형 값은 숫자 상수, 색상 리터럴, 날짜-시간 리터럴, 문자 상수 및 열거값으로도 표시될 수 있습니다.

추가 정수 유형 별칭 (C/C ++ 호환성)

C 및 C ++와 같은 다른 언어의 코드 포팅을 단순화하기 위해 표준 정수 유형에 대한 별칭이 MQL5 언어에 추가되었습니다. 이러한 별칭은 새로운 유형을 소개하지 않지만 MQL5의 기존 유형의 별칭이며 기본 유형이 적용 가능한 모든 상황에서 사용할 수 있습니다.

타입	MQL5 타입과 비교	Size in Bytes	Sign	Minimum Value	Maximum Value
int8_t	char	8	signed	-128	127

© 2000-2025, MetaQuotes Ltd.

타입	MQL5 타입과 비교	Size in Bytes	Sign	Minimum Value	Maximum Value
uint8_t	uchar	8	unsigned	0	255
int16_t	short	16	signed	-32 768	32 767
uint16_t	ushort	16	unsigned	0	65 535
int32_t	int	32	signed	- 2 147 483 648	2 147 483 647
uint32_t	uint	32	unsigned	0	4 294 967 295
int64_t	long	64	signed	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
uint64_t	ulong	64	unsigned	0	18 446 744 073 709 551 615

이러한 별명은 기존 라이브러리 및 알고리즘을 C/C ++에서 MQL5로, 특히 플랫폼 호환성이 높은 프로젝트에서 포팅 할 때 유용합니다. 모든 새로운 별명은 스크립트, EA 및 지표에서 찾아볼수 있습니다.

더 보기

전환 함수, 숫자 유형 상수

© 2000-2025, MetaQuotes Ltd.

Char, Short, Int 및 Long 타입

char

char 유형은 1바이트의 메모리(8비트)를 사용하며 이진 표기법 2^8=256 값으로 표현할 수 있습니다. *char* 유형에는 양의 값과 음의 값이 모두 포함될 수 있습니다. 값의 범주는 -128 부터 127 까지 입니다.

uchar

uchar 정수 타입도 *char* 타입뿐만 아니라 1바이트의 메모리를 차지하지만 *uchar*는 이와 달리 양의 값에만 사용됩니다. 최소값은 0이고 최대값은 255입니다. *uchar* 유형의 이름에서 첫번째 문자 u는 *unsigned*의 약자입니다.

short

short 유형의 크기는 2바이트(16비트)이므로 2와 같은 값 범위를 전력 16: 2^16 = 65 536으로 표시할 수 있습니다.*short* 유형은 부호화된 유형이며 양의 값과 음의 값을 모두 포함하므로 값의 범위는 -32 768에서 32 767 사이입니다.

ushort

부호 없는 *short* 타입은 크기가 2바이트인 *ushort* 유형입니다. 최소값은 0, 최대값은 65 535 입니다.

int

int 유형의 크기는 4바이트(32비트)입니다. 최소값은 -2 147 483 648, 최대값은 2 147 483 647 입니다.

uint

부호 없는 정수 유형은 *uint* 입니다. 4바이트의 메모리가 소요되며 0부터 4 294 967 295 까지의 정수를 표현할 수 있습니다.

long

long 타입의 크기는 8바이트 (64비트)입니다. 최소값은 -9 223 372 036 854 775 808, 최대값은 9 223 372 036 854 775 807 입니다.

ulong

ulong 타입은 또한 8바이트를 차지하며 값을 0부터 18 446 744 073 709 551 615까지 저장할 수 있습니다.

예제:

```
char  ch=12;
short sh=-5000;
int   in=2445777;
```

부호 없는 정수 유형은 음수 값을 저장하도록 설계되지 않았기 때문에 음수 값을 설정하려고 하면 예기치 않은 결과가 발생할 수 있습니다. 이러한 간단한 스크립트는 무한 루프(loop)로 이어집니다:

© 2000-2025, MetaQuotes Ltd.

```
//--- 무한 루프
void OnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<128;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
    }
  }
```

올바른 변수는:

```
//--- 올바른 변수
void OnStart()
  {
   uchar  u_ch;

   for(char ch=-128;ch<=127;ch++)
     {
      u_ch=ch;
      Print("ch = ",ch," u_ch = ",u_ch);
      if(ch==127) break;
    }
  }
```

결과:

```
   ch= -128  u_ch= 128
   ch= -127  u_ch= 129
   ch= -126  u_ch= 130
   ch= -125  u_ch= 131
   ch= -124  u_ch= 132
   ch= -123  u_ch= 133
   ch= -122  u_ch= 134
   ch= -121  u_ch= 135
   ch= -120  u_ch= 136
   ch= -119  u_ch= 137
   ch= -118  u_ch= 138
   ch= -117  u_ch= 139
   ch= -116  u_ch= 140
   ch= -115  u_ch= 141
   ch= -114  u_ch= 142
   ch= -113  u_ch= 143
   ch= -112  u_ch= 144
   ch= -111  u_ch= 145
    ...
```

예제:

© 2000-2025, MetaQuotes Ltd.

```
//--- 부호가 없는 유형에는 음수 값을 저장할 수 없습니다
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

16진수: 숫자 0-9, 10-15의 a-f 또는 A-F 문자, 0x 또는 0X로 시작합니다.

예제:

```
0x0A, 0x12, 0X12, 0x2f, 0xA3, 0Xa3, 0X7C7
```

정수 변수의 경우 값을 B 접두사를 사용하여 이진 형식으로 설정할 수 있습니다. 예를 들어, 거래 세션의 작업 시간을 int 유형 변수로 인코딩하고 필요한 알고리즘에 따라 거래 세션에 대한 정보를 사용할 수 있습니다:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 근무 시간에 대해 1을 설정하고 근무하지 않는 시간에 대해 0을 설정합니다
   int AsianSession   =B'111111111'; // 아시아 세션 0:00 ~ 9:00
   int EuropeanSession=B'111111110000000000'; // 유럽 세션 9:00 - 18:00
   int AmericanSession =B'111111110000000000000011'; // 미국 세션 16:00 - 02:00
//--- 세션의 수치 값 구하기
   PrintFormat("아시아 세션 시간의 값은 =%d",AsianSession);
   PrintFormat("유럽 세션 시간의 값은 %d",EuropeanSession);
   PrintFormat("미국 세션 시간의 값은 %d",AmericanSession);
//--- 이제 세션의 작업 시간에 대한 문자열 표현을 표시하겠습니다
   Print("아시아 세션 ",GetHoursForSession(AsianSession));
   Print("유럽 세션 ",GetHoursForSession(EuropeanSession));
   Print("미국 세션 ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------+
//| 세션의 작업 시간을 문자열로 반환                                 |
//+------------------------------------------------------------------+
string GetHoursForSession(int session)
  {
//--- 확인하려면, AND 비트 연산 및 1 비트 <<=1 의 좌측 시프트를 사용하십시오
//--- 가장 낮은 비트부터 검사를 시작합니다
   int bit=1;
   string out="working hours: ";
//--- 0에서 시작하여 최대 23까지 24비트를 모두 검사합니다.
   for(int i=0;i<24;i++)
     {
      //--- 수의 비트 상태 수신
      bool workinghour=(session&bit)==bit;
      //--- 메시지에 시간 수를 추가
      if(workinghour )out=out+StringFormat("%d ",i);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 왼쪽으로 한 비트씩 이동하여 다음 값 확인
      bit<<=1;
    }
//--- 결과 문자열
   return out;
  }
```

더 보기

타입캐스팅

© 2000-2025, MetaQuotes Ltd.

문자 상수

MQL5에서 문자열의 요소로 사용되는 문자는 유니코드 문자 집합의 인덱스입니다. 이 값은 정수로 캐스팅할 수 있는 16진수 값이며, 더하기 및 빼기와 같은 정수 연산으로 조작할 수 있습니다.

따옴표 또는 '₩10'으로 표시된 문자의 16진수 ASCII 코드는 문자 상수이며 ushort 타입입니다. 예를 들어, '0' 유형의 레코드는 숫자 값 30으로, 문자 테이블의 인덱스에 해당합니다.

예제:

```
void OnStart()
  {
//--- 문자 상수 정의
   int symbol_0='0';
   int symbol_9=symbol_0+9; // 심볼 '9' 획득
//--- 상수 출력 값
   printf("In a decimal form: symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("In a hexadecimal form: symbol_0 = 0x%x,  symbol_9 = 0x%x",symbol_0,symbol_9
//--- 문자열에 상수 입력
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- 이것이 그들이 문자열로 된 모습입니다
   Print(test);
  }
```

백슬래시는 프로그램의 원본 텍스트에서 상수 문자열과 문자 상수를 처리할 때 컴파일러의 제어 문자입니다. 작은따옴표('), 큰따옴표("), 백슬래시(₩)및 컨트롤 문자와 같은 일부 기호는 아래 표에 따라 백슬래시(₩)로 시작하는 기호의 조합으로 표시될 수 있습니다:

문자 이름	니모닉 코드 또는 이미지	MQL5 레코드	숫자 값
줄바꿈	LF	'\n'	10
horizontal tab	HT	'\t'	9
carriage return	CR	'\r'	13
백슬래시	\	'\\'	92
작은따옴표	'	'\''	39
큰따옴표	"	'\"'	34
16진수 코드	hhhh	'\xhhhh'	1에서 4까지의 16진수 문자
십진수 코드	d	'\d'	0 에서부터 65535 까지의 십진수

백슬래시 뒤에 위에서 설명한 문자 이외의 문자가 있으면 결과가 정의되지 않습니다.

예제

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- 문자 상수 선언
   int a='A';
   int b='$';
   int c='©';       // 코드 0xA9
   int d='\xAE';    // 심볼 ® 코드
//--- 출력 인쇄 상수
   Print(a,b,c,d);
//--- 문자열에 문자 추가
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- 문자열의 문자 교체
   StringSetCharacter(test,0,b);
   Print(test);
//--- 문자열의 문자 교체
   StringSetCharacter(test,0,c);
   Print(test);
//--- 문자열의 문자 교체
   StringSetCharacter(test,0,d);
   Print(test);
//--- 문자를 숫자로 표현
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- 문자열에 문자 추가
   StringSetCharacter(test,1,a1);
   Print(test);
//--- 문자열에 문자 추가
   StringSetCharacter(test,1,b1);
   Print(test);
//--- 문자열에 문자 추가
   StringSetCharacter(test,1,c1);
   Print(test);
//--- 문자열에 문자 추가
   StringSetCharacter(test,1,d1);
   Print(test);
  }
```

위에서 언급한 바와 같이 문자 상수(또는 변수) 값은 문자 표의 색인입니다. 인덱스는 정수이므로 다른 방법으로 쓸 수 있습니다.

```
void OnStart()
  {
//---
   int a=0xAE;      // ®의 코드는 '₩xAE' 리터럴에 해당합니다
   int b=0x24;      // $ 코드는 '₩x24' 리터럴에 해당합니다
```

© 2000-2025, MetaQuotes Ltd.

```
   int c=0xA9;      // ©의 코드는 '₩xA9' 리터럴에 해당합니다
   int d=0x263A;    // ☺ 코드는 '₩x263A' 리터럴에 해당합니다
//--- 값 보기
   Print(a,b,c,d);
//--- 문자열에 문자 추가
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- 문자열의 문자 교체
   StringSetCharacter(test,0,b);
   Print(test);
//--- 문자열의 문자 교체
   StringSetCharacter(test,0,c);
   Print(test);
//--- 문자열의 문자 교체
   StringSetCharacter(test,0,d);
   Print(test);
//--- 코드 세트
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- 스페이드 문자 추가
   StringSetCharacter(test,1,a1);
   Print(test);
//--- 하트 문자 추가
   StringSetCharacter(test,2,b1);
   Print(test);
//--- add a character of diamonds
   StringSetCharacter(test,3,c1);
   Print(test);
//--- 다이아몬드 문자 추가
   StringSetCharacter(test,4,d1);
   Print(test);
//--- 문자열의 문자 리터럴 예제
   test="Queen\x2660Ace\x2662";
   printf("%s",test);
  }
```

문자 리터럴의 내부 표현은 ushort 타입입니다. 문자 상수는 0에서 65535 사이의 값을 사용할 수 있습니다.

더 보기

StringSetCharacter(), StringGetCharacter(), ShortToString(), ShortArrayToString(), StringToShortArray()

© 2000-2025, MetaQuotes Ltd.

Datetime 타입

datetime 타입은 1970년 1월 1일 이후 경과된 시간(초)으로 날짜와 시간을 저장하기 위한 것입니다. 이 유형은 8바이트의 메모리를 차지합니다.

날짜 및 시간 상수는 리터럴 문자열로 나타낼 수 있으며, 리터럴 문자열은 연도, 월, 일(또는 일, 월, 연도), 시간, 분 및 초의 숫자 값을 보여 주는 6개 부분으로 구성됩니다. 상수는 작은따옴표로 묶이고 D 문자로 시작합니다.

값은 1970년 1월 1일부터 3000년 12월 31일까지입니다. 날짜(년, 월, 일) 또는 시간(시간, 분, 초) 또는 모두 생략할 수 있습니다.

리터럴 날짜 지정의 경우 연도, 월 및 일을 지정하는 것이 좋습니다. 그렇지 않으면 컴파일러는 불완전한 항목에 대한 경고를 반환합니다.

예제:

```
datetime NY=D'2015.01.01 00:00';      // 2015년 시작 시점
datetime d1=D'1980.07.19 12:30:27';   // 연도 월 일 시간 분 초
datetime d2=D'19.07.1980 12:30:27';   // D'1980.07.19 12:30:27' 와 같음;
datetime d3=D'19.07.1980 12';         // D'1980.07.19 12:00:00' 와 같음
datetime d4=D'01.01.2004';            // D'01.01.2004 00:00:00' 와 같음
datetime compilation_date=__DATE__;              // 편성일자
datetime compilation_date_time=__DATETIME__;     // 편성일시
datetime compilation_time=__DATETIME__-__DATE__;// 편성시간
//--- 컴파일러 경고가 반환되는 선언의 예
datetime warning1=D'12:30:27';        // D'[date of compilation] 12:30:27' 와 같음
datetime warning2=D'';                // __DATETIME__ 와 같음
```

더 보기

데이터 타입 구조, 날짜 및 시간, TimeToString, StringToTime

© 2000-2025, MetaQuotes Ltd.

Color 타입

color 타입은 색에 대한 정보를 저장하기 위한 것으로 메모리에서 4바이트를 차지합니다. 첫 번째 바이트는 무시되고 나머지 3바이트에는 RGB 구성요소가 포함됩니다.

색상 상수는 문자 그대로, 정수 또는 이름(웹-색상에만 해당)의 세 가지 방법으로 나타낼 수 있습니다.

리터럴 표현은 세 가지 주요 색상 성분의 숫자 속도 값(빨간색, 녹색, 파란색)을 나타내는 세 부분으로 구성됩니다. 상수는 C로 시작하고 작은따옴표로 묶입니다. 색상 구성요소의 숫자 속도 값은 0에서 255 사이의 범위에 있습니다.

정수 값 표현은 16진수 또는 십진수 형식으로 작성됩니다. 16진수 숫자는 0x00BBGGRR 처럼 보입니다. 여기서 RR은 빨간색 구성 요소의 비율이고 GG는 녹색 구성 요소의 비율이며 BB는 파란색 구성 요소의 비율입니다. 10진수 상수는 RGB에 직접 반영되지 않습니다. 16진수 정수 표현의 십진수 값을 나타냅니다.

특정 색상은 이른바 웹-색상 세트를 반영합니다.

예제:

```
//--- 문자 그대로
C'128,128,128'    // 회색
C'0x00,0x00,0xFF' // 파랑
//색상명
clrRed               // 빨강
clrYellow            // 노랑
clrBlack             // 검정
//--- 통합 표현
0xFFFFFF          // 흰색
16777215          // 흰색
0x008000          // 녹색
32768             // 녹색
```

더 보기

웹 색상, ColorToString, StringToColor, Typecasting

© 2000-2025, MetaQuotes Ltd.

Bool 타입

bool 타입은 참 또는 거짓의 논리 값을 저장하기 위한 것이며, 이 논리 값의 숫자 표현은 각각 1 또는 0입니다.

예제:

```
bool a = true;
bool b = false;
bool c = 1;
```

내부 표현은 정수 1바이트입니다. 논리식에서는 이러한 유형의 다른 정수 또는 실제 형식 또는 표현식을 사용할 수 있으며 컴파일러는 오류를 생성하지 않습니다. 이 경우 0 값은 거짓으로 해석되고 다른 모든 - 값은 참으로 해석됩니다.

예제:

```
   int i=5;
   double d=-2.5;
   if(i) Print("i = ",i," 그리고 참으로 설정됨");
   else Print("i = ",i," 그리고 거짓으로 설정됨");

   if(d) Print("d = ",d," 그리고 참 값을 가짐");
   else Print("d = ",d," 그리고 거짓 값을 가짐");

   i=0;
   if(i) Print("i = ",i," 그리고 참 값을 가짐");
   else Print("i = ",i," 그리고 거짓 값을 가짐");

   d=0.0;
   if(d) Print("d = ",d," 그리고 참 값을 가짐");
   else Print("d = ",d," 그리고 거짓 값을 가짐");

//--- 실행 결과
//    i = 5 이고 참 값을 가짐
//    d= -2.5 이고 참 값을 가짐
//    i = 0 이고 거짓 값을 가짐
//    d= 0 이고 거짓 값을 가짐
```

더 보기

Boolean 연산, 우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

열거

enum 유형의 데이터는 제한된 데이터 집합에 속합니다. 열거 유형 정의:

```
enum 명칭, 열거 가능한 유형의
  {
  값 목록
  };
```

값 목록은 쉼표로 구분된 명명된 상수의 식별자 목록입니다.

예제:

```
enum 개월  // 명명된 상수 열거
   {
   1월,
   2월,
   3월,
   4월,
   5월,
   6월,
   7월,
   8월,
   9월,
   10월,
   11월,
   12월
   };
```

열거가 선언되면 새 정수 값 4바이트 데이터 유형이 나타납니다. 열거에는 이름이 지정된 새 상수가 적용되므로 새 데이터 유형을 선언하면 컴파일러에서 전달된 매개 변수의 유형을 엄격하게 제어할 수 있습니다. 위의 예제에서 1월 명명 상수의 값은 0, 2월 -1, 12월 -11입니다.

규칙: 특정 값이 열거형 멤버인 명명된 상수에 할당되지 않으면 새 값이 자동으로 형성됩니다. 열거의 첫 번째 멤버인 경우 0 값이 할당됩니다. 이후의 모든 멤버에 대해 이전 멤버의 값에 따라 값을 계산하여 하나를 추가합니다.

예제:

```
enum intervals  // 명명된 상수 열거
   {
    month=1,     // 1개월 간격
    two_months,  // 2개월
    quarter,     // 3개월 - 1분기
    halfyear=6,  // 반년
    year=12,     // 1년
   };
```

참고

- C++와 달리 MQL5에서 열거된 유형의 내부 표현 크기는 항상 4바이트입니다. 즉, sizeof (개월)은 4를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

- C++와 달리 MQL5에서는 익명 열거를 선언할 수 없습니다. 즉, 항상 enum 키워드 뒤에 고유 이름을 지정해야 합니다.

더 보기

타입캐스팅

© 2000-2025, MetaQuotes Ltd.

실수 유형 (double, float)

실수 유형(또는 부동 소수점 유형)은 분수 부분이 있는 값을 나타냅니다. MQL5 언어에는 부동 소수점 번호에 대한 두 가지 유형이 있습니다.컴퓨터 메모리의 실수 표시 방법은 IEEE 754 표준에 의해 정의되며 플랫폼, 운영 체제 또는 프로그래밍 언어와 독립적입니다.

유형	크기 (바이트)	최소 양수 값	최소값	C++ 아날로그
float	4	1.175494351e-38	3.402823466e+38	float
double	8	2.2250738585072014e-308	1.7976931348623158e+308	double

double

double 실수형은 64비트(부호 1비트, 지수 11비트, 가수 52비트)를 차지합니다.

float

float 실수형은 32비트(부호 1비트, 지수 8비트, 가수 23비트)를 차지합니다.

벡터

더블 타입 숫자의 의 1차원 배열. 데이터 메모리는 동적으로 할당됩니다. 벡터 속성은 다음을 사용하여 얻을 수 있습니다.methods, 이때 벡터 크기는 변경할 수 있습니다. 벡터<double> 항목은 템플릿 함수에서 사용될 수 있습니다.

vectorf

1차원 배열 float 유형 숫자가 정밀도 손실이 문제 되지 않을 경우 벡터 대신에 사용될 수 있습니다. 벡터<float> 항목은 템플릿 함수에서 사용할 수 있습니다.

vectorc

복소수 유형의 수의 1차원 배열은 복소수를 처리하기 위한 것입니다. 벡터<complex> entry는 템플릿 함수에서 사용할 수 있습니다. vectorc 유형의 벡터 작업은 아직 구현되지 않았습니다.

행렬

행렬은 2차원 배열로 double 타입 숫자의 배열입니다. 행렬 요소의 메모리는 동적으로 분배됩니다. 행렬 속성은 다음을 사용하여 얻을 수 있습니다. method, 이때 행렬 모양은 변경될 수 있습니다. 매트릭스<double> entry는 템플릿 함수에서 사용할 수 있습니다.

matrixf

© 2000-2025, MetaQuotes Ltd.

2차원 배열 float 유형 숫자가 정밀도 손실이 문제 되지 않을 경우 행렬 대신에 사용될 수 있습니다. 행렬<float> entry는 템플릿 함수에서 사용할 수 있습니다.

matrixc

복소수 유형의 수의 1차원 배열은 복소수를 처리하기 위한 것입니다. 매트릭스<complex> entry는 템플릿 함수에서 사용할 수 있습니다. matrixc 유형의 벡터 작업은 아직 구현되지 않았습니다.

double 명칭은 이 숫자들의 정확도가 float 유형의 숫자보다 정확도가 2배 높다는 것을 의미합니다. 대부분의 경우, double 타입이 가장 편리합니다. 대부분의 경우 제한된 정밀도인 float 숫자만으로는 충분하지 않습니다. float 유형이 여전히 사용되는 이유는 메모리를 저장하기 때문입니다(실수의 큰 배열에 중요함).

부동 소수점 상수는 정수 부분, 점(.) 및 분수 부분으로 구성됩니다. 정수 및 분수 부분은 소수 자릿수의 시퀀스입니다.

예제:

```
double a=12.111;
double b=-956.1007;
float  c =0.0001;
float  d =16;
```

실수 상수를 쓰는 과학적인 방법이 있는데, 종종 이러한 기록 방법은 전통적인 것보다 더 간결합니다.

예제:

```
double c1=1.12123515e-25;
double c2=0.00000000000000000000000112123515; // 소수점 뒤의 24개의 0

Print("1. c1 =",DoubleToString(c1,16));
// 결과: 1. c1 = 0.0000000000000000

Print("2. c1 =",DoubleToString(c1,-16));
// 결과: 2. c1 = 1.1212351499999999e-025

Print("3. c2 =",DoubleToString(c2,-16));
// 결과: 3. c2 = 1.1212351499999999e-025
```

실수들은 2진수 체계에서 어느 정도 제한된 정확도로 메모리에 저장되지만, 일반적으로 소수 표기법이 사용된다는 점을 기억해야 합니다. 그렇기 때문에 십진법으로 정확하게 표현되는 많은 숫자들은 이진법에서 무한분수로만 쓸 수 있습니다.

예를 들어, 0.3과 0.7은 컴퓨터에서 무한 분수로 표시되는 반면 0.25는 두 개의 검정력을 나타내기 때문에 정확히 저장됩니다.

이와 관련하여, 동일성에 대해 두 실수 비교는 정확하지 않기 때문에 비교하지 않는 것이 강력히 권장됩니다.

예제:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//---
   double three=3.0;
   double x,y,z;
   x=1/three;
   y=4/three;
   z=5/three;
   if(x+y==z)
      Print("1/3 + 4/3 == 5/3");
   else
      Print("1/3 + 4/3 != 5/3");
// 결과: 1/3 + 4/3 != 5/3
  }
```

여전히 두 실수의 동일성을 비교해야 하는 경우 두 가지 다른 방법으로 이 작업을 수행할 수 있습니다. 첫 번째 방법은 두 숫자의 차이를 비교 정확도를 지정하는 소량으로 비교하는 것입니다.

예제:

```
bool EqualDoubles(double d1,double d2,double epsilon)
  {
   if(epsilon<0)
      epsilon=-epsilon;
//---
   if(d1-d2>epsilon)
      return false;
   if(d1-d2<-epsilon)
      return false;
//---
   return true;
  }
void OnStart()
  {
   double d_val=0.7;
   float  f_val=0.7;
   if(EqualDoubles(d_val,f_val,0.000000000000001))
      Print(d_val," equals ",f_val);
   else
      Print("Different: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToStrinc
// 결과: Different: d_val = 0.7000000000000000   f_val = 0.6999999880790710
  }
```

위의 예에서 엡실론 값은 사전 정의된 상수 DBL_EPSILON보다 작을 수 없습니다. 이 상수의 값은 2.2204460492503131e-016 입니다. float dbguddp 해당하는 상수는 FLT_EPSILON = 1.192092896e-07 입니다. 이 값의 의미는 다음과 같습니다: 다음 조건을 만족하는 가장 낮은 값입니다 1.0 + DBL_EPSILON! = 1.0 (for numbers of float type 1.0 + FLT_EPSILON! = 1.0).

© 2000-2025, MetaQuotes Ltd.

두 번째 방법은 두 실수 간의 정규화된 차이를 0과 비교하는 방법을 제공합니다. 정규화된 숫자를 사용하는 모든 수학 연산은 정규화되지 않은 결과를 제공하므로 정규화된 숫자의 차이를 0으로 비교하는 것은 의미가 없습니다.

예제:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0)
      return(true);
   else
      return(false);
  }
void OnStart()
  {
   double d_val=0.3;
   float  f_val=0.3;
   if(CompareDoubles(d_val,f_val))
      Print(d_val," equals ",f_val);
   else
      Print("Different: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToStrin
// 결과: Different: d_val = 0.3000000000000000   f_val = 0.3000000119209290
  }
```

수학적 보조 프로세서의 일부 연산은 잘못된 실수 수를 발생시킬 수 있으며, 이는 잘못된 실수 수를 가진 연산의 결과가 정의되지 않기 때문에 수학적 연산 및 비교 연산에 사용할 수 없습니다. 예를 들어, 2의 아크사인을 계산하려고 하면 음의 무한대가 됩니다.

예제:

```
   double abnormal = MathArcsin(2.0);
   Print("MathArcsin(2.0) =",abnormal);
// 결과:  MathArcsin(2.0) = -1.#IND
```

마이너스 무한대 외에 플러스 무한대와 NaN(숫자가 아님)이 있습니다. 이 숫자가 유효하지 않은지 확인하려면 MathIsValidNumber()를 사용하면 됩니다. IEEE 표준에 따르면, 그들은 특별한 기계 표현을 가지고 있습니다. 예를 들어, 이중 유형에 대한 더하기 무한대의 비트 표현은 0x7FF0 0000 0000 0000입니다.

예제:

```
struct str1
  {
   double d;
  };
struct str2
  {
   long l;
  };

//--- 시작
   str1 s1;
```

© 2000-2025, MetaQuotes Ltd.

```
   str2 s2;
//---
   s1.d=MathArcsin(2.0);         // 유효하지 않은 숫자 가져오기 -1.#IND
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;      // 유효하지 않은 숫자 -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;      // 가장 큰 비-숫자 SNaN
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;      // 가장 작은 비-숫자 QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;      // 가장 큰 비-숫자 QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;      // 양의 무한대 1.#INF 및 가장 작은 비-숫자 SNaN
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;      // 음의 무한대 -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;      // 음수 영 -0.0
   s1=s2;
   printf("8.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FE0000000000000;      // 0.5
   s1=s2;
   printf("9.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x3FF0000000000000;      // 1.0
   s1=s2;
   printf("10.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FEFFFFFFFFFFFFF;      // 최대 정규화 수 (MAX_DBL)
   s1=s2;
   printf("11.  %.16e %I64X",s1.d,s2.l);
//---
   s2.l=0x0010000000000000;      // 최소 양의 정규화 (MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   s1.d=0.7;                     // 0.7의 수가 무한 분수임을 증명하라
   s2=s1;
   printf("13.  %.16e %.16I64X",s1.d,s2.l);
/*
1.  -1.#IND00 FFF8000000000000
2.  -1.#QNAN0 FFFF000000000000
3.   1.#SNAN0 7FF7000000000000
4.   1.#QNAN0 7FF8000000000000
5.   1.#QNAN0 7FFF000000000000
6.   1.#INF00 7FF0000000000000
7.  -1.#INF00 FFF0000000000000
8.  -0.000000 8000000000000000
9.   0.500000 3FE0000000000000
10.  1.000000 3FF0000000000000
11.  1.7976931348623157e+308 7FEFFFFFFFFFFFFF
12.  2.2250738585072014e-308 0010000000000000
13.  6.9999999999999996e-001 3FE6666666666666
*/
```

더 보기

DoubleToString, NormalizeDouble, Numeric Type Constants

© 2000-2025, MetaQuotes Ltd.

복소수(complex)

내장 복잡한 유형은 두 개의 double 필드가 있는 구조입니다:

```
struct complex
  {
   double            real;    // Real part
   double            imag;    // Imaginary part
  };
```

"complex" 유형은 value에 의해 MQL5 기능의 매개변수로 전달될 수 있습니다(참조로만 전달되는 일반 구조와 대조적으로). DLL에서 가져온 함수의 경우 "complex" 유형은 참조로만 전달되어야 합니다.

'i' suffix는 complex 상수를 설명하는 데 사용됩니다.

```
complex square(complex c)
  {
   return(c*c);
  }
void OnStart()
  {
   Print(square(1+2i));  // 상수가 매개변수로 전달되었습니다
  }
// "(-3, 4)" 가 복소수의 스트링으로 전달되었습니다.
```

현재 복소수에는 단순 연산자만 사용할 수 있습니다: =, +, -, *, /, +=, -=, *=, /=, ==,!=.

추가 수학 함수에 대한 지원은 나중에 추가 될 예정입니다. 절대값, 사인, 코사인 등을 계산할 수 있을 것입니다.

vectorc

복소수 유형의 수의 1차원 배열은 복소수를 처리하기 위한 것입니다. 벡터<complex> entry는 템플릿 함수에서 사용할 수 있습니다. vectorc 유형의 벡터 작업은 아직 구현되지 않았습니다.

행렬

행렬은 2차원 배열로 double 타입 숫자의 배열입니다. 행렬 요소의 메모리는 동적으로 분배됩니다. 행렬 속성은 다음을 사용하여 얻을 수 있습니다. method, 이때 행렬 모양은 변경될 수 있습니다. 매트릭스<double> entry는 템플릿 함수에서 사용할 수 있습니다.

matrixf

2차원 배열 float 유형 숫자가 정밀도 손실이 문제 되지 않을 경우 행렬 대신에 사용될 수 있습니다. 행렬<float> entry는 템플릿 함수에서 사용할 수 있습니다.

matrixc

복소수 유형의 수의 1차원 배열은 복소수를 처리하기 위한 것입니다. 매트릭스<complex> entry는 템플릿 함수에서 사용할 수 있습니다. matrixc 유형의 벡터 작업은 아직 구현되지 않았습니다.

© 2000-2025, MetaQuotes Ltd.

문자열 유형

문자열 유형은 텍스트 문자열을 저장하는 데 사용됩니다. 텍스트 문자열은 유니코드 형식의 문자 시퀀스이며 끝에는 최종 0이 있습니다. 문자열 상수를 문자열 변수에 할당할 수 있습니다. 문자열 상수는 큰따옴표로 묶인 유니코드 문자의 시퀀스입니다: "이것은 문자열 상수입니다."

큰따옴표(")를 문자열에 포함해야 하는 경우, 백슬래시 문자(₩)를 문자열 앞에 넣어야 합니다. 백슬래시 문자(₩)를 앞에 입력하면 문자열에 특수 문자 상수를 쓸 수 있습니다.

예제:

```
string svar="이것은 문자 상수입니다";
string svar2=StringSubstr(svar,0,4);
Print("Copyright symbol\t\x00A9");
FileWrite(handle,"이 문자열에는 줄바꿈 기호가 포함됩니다 ₩n");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

소스 코드를 읽을 수 있도록 추가 작업 없이 긴 상수 문자열을 여러 부분으로 분할할 수 있습니다. 컴파일하는 동안 이러한 파트들은 하나의 긴 문자열로 결합됩니다:

```
//--- long 상수 문자열 선언
   string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\""
                    " \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n"
                    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
                    "<head>\n"
                    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ut
                    "<title>Trade Operations Report</title>\n"
                    "</head>";
//--- 상수 문자열을 로그에 출력합니다
   Print(HTML_head);
  }
```

더 보기

함수 전환, 문자열 함수, FileOpen, FileReadString, FileWriteString

© 2000-2025, MetaQuotes Ltd.

구조, 클래스 및 인터페이스

구조

구조는 모든 유형의 요소의 집합입니다(void 유형 제외). 따라서 구조는 서로 다른 유형의 논리적으로 관련된 데이터를 결합합니다.

구조 선언

구조 데이터 유형은 다음 설명에 따라 결정됩니다:

```
struct structure_name
  {
   elements_description
  };
```

구조명을 식별자(변수 또는 함수의 이름)로 사용할 수 없습니다. MQL5에서 구조 요소는 정렬 없이 서로 직접 따릅니다. C++에서는 다음 명령을 사용하여 컴파일러에 이러한 순서가 지정됩니다:

```
#pragma pack(1)
```

구조에서 다른 정렬을 수행하려면 적절한 크기로 보조 멤버인 "채우기"를 사용하십시오.

예제:

```
struct trade_settings
  {
   uchar  slippage;     // 허용되는 저하 크기 1바이트 값
   char   reserved1;    // 1바이트 건너뛰기
   short  reserved2;    // 2바이트 건너뛰기
   int    reserved4;    // 또 다른 4바이트 건너뛰기. 경계 8바이트 정렬 확인
   double take;         // 차익고정가격의 가치
   double stop;         // 보호용 스톱의 가격 가치
  };
```

정렬된 구조에 대한 이러한 설명은 가져온 dll-함수로 전송하는 경우에만 필요합니다.

주의: 이 예에서는 잘못 설계된 데이터를 보여 줍니다. 먼저 *take* 및 *stop* 대용량 데이터를 double 유형으로 선언한 다음, *slippage* 멤버를 uchar 타입으로 선언하는 것이 좋습니다. 이 경우 데이터의 내부 표현은 *#pragma pack()*에 지정된 값에 관계없이 항상 동일합니다.

구조에 string 유형 및/또는 동적 배열의 객체 변수가 포함된 경우 컴파일러는 이러한 구조에 암시적 생성자를 할당합니다. 이 생성자는 string 유형의 모든 구조 멤버를 재설정하고 동적 배열의 객체를 올바르게 초기화합니다.

단순 구조

동적 배열의 문자열, 클래스 객체, 포인터 및 객체를 포함하지 않는 구조를 단순 구조체라고 합니다. 단순 구조의 변수와 해당 배열은 DLL에서 임포트한 함수에 매개 변수로 전달될 수 있습니다.

단순 구조의 복사는 두 가지 경우에만 허용됩니다:

- 객체가 동일한 구조 유형에 속하는 경우

© 2000-2025, MetaQuotes Ltd.

- 한 구조가 다른 구조의 후손임을 의미하는 계통에 의해 객체가 연결되어 있는지 여부.

예를 들면, CustomMqlTick 커스텀 구조를 내장된 MqlTick와 동일한 내용으로 개발해봅시다. 컴파일러에서는 MqlTick 개체 값을 CustomMqlTick 유형 개체로 복사할 수 없습니다. 다이렉트 타입캐스팅-필요한 유형에 대한-도 컴파일 오류를 발생시킵니다:

```
      //--- 다른 유형의 단순 구조를 복사할 수 없습니다
      my_tick1=last_tick;               // 컴파일러는 여기에 오류를 반환합니다

      //--- 서로 다른 유형의 타입캐스팅 구조도 금지됩니다.
      my_tick1=(CustomMqlTick)last_tick;// 컴파일러는 여기에 오류를 반환합니다
```

따라서 구조 요소의 값을 하나씩 복사하는 옵션만 남아 있습니다. 동일한 유형의 CustomMqlTick 값을 복사할 수 있습니다.

```
      CustomMqlTick my_tick1,my_tick2;
      //--- 동일한 유형의 CustomMqlTick 개체를 다음과 같이 복사할 수 있습니다
      my_tick2=my_tick1;

      //--- 단순 CustomMqlTick 구조의 개체에서 배열을 만들고 여기에 값을 씁니다
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
```

ArrayPrint() 함수는 저널에 *arr[] 배열 값을 표시하기 위해 호출됩니다.*

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 내장된 MqlTick과 유사한 구조를 개발합니다
   struct CustomMqlTick
     {
      datetime          time;          // 최종가 업데이트 시간
      double            bid;           // 현재 입찰가
      double            ask;           // 현재 요청가
      double            last;          // 마지막 거래의 현재 가격
      ulong             volume;        // 현재 최종가의 거래량
      long              time_msc;      // 최종 가격 업데이트 시간(밀리초)
      uint              flags;         // 틱 플래그
     };
   //--- 마지막 틱 값 가져오기
   MqlTick last_tick;
   CustomMqlTick my_tick1,my_tick2;
//--- MqlTick에서 CustomMqlTick으로 데이터 복사 시도
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      //--- 관련 없는 단순 구조를 복사할 수 없습니다
      //1. my_tick1=last_tick;               // 컴파일러는 여기에 오류를 반환합니다
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 서로 관련 없는 구조물을 타입캐스팅하는 것도 금지됩니다
      //2. my_tick1=(CustomMqlTick)last_tick;// 컴파일러는 여기에 오류를 반환합니다

      //--- 따라서 구조 멤버를 하나씩 복사합니다.
      my_tick1.time=last_tick.time;
      my_tick1.bid=last_tick.bid;
      my_tick1.ask=last_tick.ask;
      my_tick1.volume=last_tick.volume;
      my_tick1.time_msc=last_tick.time_msc;
      my_tick1.flags=last_tick.flags;

      //--- 동일한 유형의 CustomMqlTick 개체를 다음과 같이 복사할 수 있습니다
      my_tick2=my_tick1;

      //--- 단순 CustomMqlTick 구조의 개체에서 배열을 만들고 여기에 값을 씁니다
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
      ArrayPrint(arr);
//--- CustomMqlTick 유형의 개체가 포함된 배열의 값 표시 예제
      /*
                     [시간]    [입찰]    [요청]    [최근] [거래량]     [time_msc] [flags]
      [0] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      [1] 2017.05.29 15:04:37 1.11854 1.11863 +0.00000  1450000 1496070277157       2
      */
    }
   else
      Print("SymbolInfoTick() 실패, 오류 = ",GetLastError());
  }
```

두 번째 예는 계통을 기준으로 간단한 구조를 복사하는 기능을 보여줍니다. 우리가 고양이와 개의 구조가 파생된 동물의 기본 구조를 가지고 있다고 가정해보죠. 동물과 고양이는 물론 동물과 개도 서로 복사할 수 있지만 고양이와 개는 둘 다 동물 구조의 후손이지만 서로 복사할 수는 없습니다.

```
//--- 개를 묘사하는 구조
struct Dog: Animal
  {
   bool              hunting;       // 사냥 종
  };
//--- 고양이를 묘사하는 구조
struct Cat: Animal
  {
   bool              home;          // 가축
  };
//--- 자식 구조의 객체 생성
   Dog dog;
   Cat cat;
//--- 부모 항목에서 자식 항목으로 복사할 수 있습니다. (Animal ==> Dog)
   dog=some_animal;
```

© 2000-2025, MetaQuotes Ltd.

```
   dog.swim=true;    // 강아지는 수영할수 있습니다
//--- 자식 구조의 객체를 복사할 수 없습니다 (Dog != Cat)
   cat=dog;          // 컴파일러가 오류를 반환합니다
```

전체 예제 코드:

```
//--- 동물을 묘사하는 기본 구조
struct Animal
  {
   int               head;          // 머리 수
   int               legs;          // 다리 수
   int               wings;         // 날개 수
   bool              tail;          // 꼬리
   bool              fly;           // 비행
   bool              swim;          // 수영
   bool              run;           // 달리기
  };
//--- 개를 묘사하는 구조
struct Dog: Animal
  {
   bool              hunting;       // 사냥 종
  };
//--- 고양이를 묘사하는 구조
struct Cat: Animal
  {
   bool              home;          // 가축
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 기본 동물 유형의 개체를 만들고 설명합니다
   Animal some_animal;
   some_animal.head=1;
   some_animal.legs=4;
   some_animal.wings=0;
   some_animal.tail=true;
   some_animal.fly=false;
   some_animal.swim=false;
   some_animal.run=true;
//--- 자식 유형의 개체 만들기
   Dog dog;
   Cat cat;
//--- 부모 항목에서 자식 항목으로 복사할 수 있습니다. (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // 개는 수영할 수 있습니다
//--- 자식 구조의 객체를 복사할 수 없습니다 (Dog != Cat)
   //cat=dog;          // 컴파일러가 여기서 오류를 반환합니다
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 따라서 요소를 하나씩만 복사할 수 있습니다
   cat.head=dog.head;
   cat.legs=dog.legs;
   cat.wings=dog.wings;
   cat.tail=dog.tail;
   cat.fly=dog.fly;
   cat.swim=false;   // 고양이는 수영을 못합니다
//--- 자식 항목에서 부모 항목으로 값을 복사할 수 있습니다
  동물 코끼리;
   elephant=cat;
   elephant.run=false;// 코끼리는 달릴 수 없습니다
   elephant.swim=true;// 코끼리는 수영할 수 있습니다
//--- 배열 생성
   Animal animals[4];
   animals[0]=some_animal;
   animals[1]=dog;
   animals[2]=cat;
   animals[3]=elephant;
//--- print out
   ArrayPrint(animals);
//--- 실행 결과
/*
      [머리] [다리] [날개] [꼬리] [비행] [수영] [달리기]
   [0]      1      4      0   true false  false  true
   [1]      1      4      0   true false   true  true
   [2]      1      4      0   true false  false false
   [3]      1      4      0   true false   true false
*/
  }
```

간단한 유형을 복사하는 또 다른 방법은 유니언을 사용하는 것입니다. 구조물의 대상은 동일한 조합원이어야 합니다 - 조합의 예를 참조하십시오.

구조 멤버에 대한 액세스

구조명이 새 데이터 유형이 되므로 이 유형의 변수를 선언할 수 있습니다. 구조는 프로젝트 내에서 한 번만 선언할 수 있습니다. 구조 멤버는 다음을 사용하여 액세스합니다:포인트 연산 (.).

예제:

```
struct trade_settings
  {
   double take;         // 차익 확정 가격의 가치
   double stop;         // 보호 중지 가격의 가치
   uchar  slippage;     // 허용 저하 값
  };
//--- trade_settings 변수 생성 및 초기화
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

© 2000-2025, MetaQuotes Ltd.

구조 및 클래스 필드 정렬을 위한 'pack'

특수 pack 속성을 사용하면 구조 또는 클래스 필드의 정렬을 설정할 수 있습니다.

```
pack([n])
```

여기서 n은 다음 값 중 하나입니다: 1, 2, 4, 8 또는 16. 없을 수도 있습니다.

예제:

```
   struct pack(sizeof(long)) MyStruct
     {
      // 구조 멤버는 8바이트 경계에 맞춰져야 합니다
     };
또는
   struct MyStruct pack(sizeof(long))
     {
      // 구조 멤버는 8바이트 경계에 맞춰져야 합니다
     };
```

'pack(1)' 은 기본적으로 구조에 적용됩니다. 즉, 구조 멤버가 메모리에 하나씩 위치하며 구조 크기는 해당 멤버의 크기를 합한 값과 같습니다.

예제:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 정렬이 없는 단순 구조
   struct Simple_Structure
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- 단순 구조 인스턴스 선언
   Simple_Structure s;
//--- 각 구조 멤버의 크기 표시
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- POD 구조의 크기가 멤버의 크기의 합과 같은지 확인합니다.
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  결과:
   sizeof(s.c)=1
   sizeof(s.s)=2
```

© 2000-2025, MetaQuotes Ltd.

```
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=15
*/
  }
```

이러한 정렬이 적용되는 써드파티 라이브러리(*.DLL)와 데이터를 교환할 때 구조 필드 정렬이 필요할 수 있습니다.

몇 가지 예를 사용하여 정렬의 작동 방식을 보여 드리겠습니다. 우리는 정렬되지 않은 4개의 멤버로 구성된 구조를 적용할 것입니다.

```
//--- 정렬이 없는 단순 구조
   struct Simple_Structure pack() // 크기가 지정되지 않았습니다. 1바이트의 경계에 대한 정렬을
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
//--- 단순 구조 인스턴스 선언
   Simple_Structure s;
```

구조 필드는 다음에 따라 차례로 메모리에 위치해야 합니다: 선언 순서 및 유형 크기. 구조 크기는 15이지만 배열의 구조 필드에 대한 오프셋은 정의되지 않았습니다.

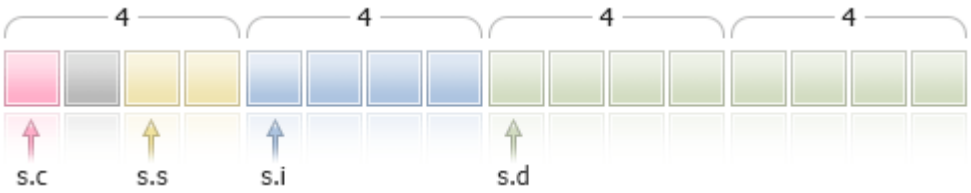
이제 4바이트 정렬과 동일한 구조를 선언하고 코드를 실행합니다.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4바이트 정렬의 단순 구조
   struct Simple_Structure pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- 단순 구조 인스턴스 선언
   Simple_Structure s;
//--- 각 구조 멤버의 크기 표시
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- POD 구조의 크기가 멤버 크기의 합과 같지 않은지 확인합니다
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  결과:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=16 // 구조 크기가 변경됨
*/
  }
```

구조 크기가 변경되어 4바이트 이상의 모든 멤버가 4바이트 배수의 구조 시작 부분부터 오프셋을 갖습니다. 작은 멤버는 자체 크기 경계에 맞춰 정렬됩니다(예: 'short'의 경우 2). 추가된 바이트는 회색으로 표시됩니다.



이 경우 **s.c** 멤버 뒤에 1바이트가 추가되므로 **s.s**(sizeof(short)==2) 필드의 경계가 2바이트('short' 타입에 대한 정렬)가 됩니다.

배열에서 구조의 시작에 대한 오프셋도 4바이트 경계에 맞춰집니다. 즉, a[0], a[1] 및 a[n] 요소의 주소는 Simple_Structure arr[]의 경우 4바이트의 배수여야 합니다.

4바이트 정렬과 멤버 순서가 다른 유사한 유형으로 구성된 두 개의 구조를 더 고려해 보겠습니다. 첫 번째 구조에서 멤버는 유형 크기 오름차순으로 위치합니다.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4바이트 경계에 정렬된 단순 구조
   struct CharShortInt pack(4)
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(double)=4
     };
//--- 단순 구조 인스턴스 선언
   CharShortInt ch_sh_in;
//--- 각 구조 멤버의 크기 표시
   Print("sizeof(ch_sh_in.c)=",sizeof(ch_sh_in.c));
   Print("sizeof(ch_sh_in.s)=",sizeof(ch_sh_in.s));
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("sizeof(ch_sh_in.i)=",sizeof(ch_sh_in.i));

//--- POD 구조의 크기가 멤버의 크기의 합과 같은지 확인합니다.
   Print("sizeof(CharShortInt)=",sizeof(CharShortInt));
/*
  결과:
   sizeof(ch_sh_in.c)=1
   sizeof(ch_sh_in.s)=2
   sizeof(ch_sh_in.i)=4
   sizeof(CharShortInt)=8
*/
  }
```

우리가 볼 수 있듯이 구조 크기는 8이며 2개의 4바이트 블록으로 구성되어 있습니다. 첫 번째 블록은 'char' 및 'short' 유형의 필드를 포함하고 두 번째 블록은 'int' 유형의 필드를 포함합니다.

이제 'short' 타입 멤버를 끝으로 이동하여 첫 번째 구조를 필드 순서만 다른 두 번째 구조로 전환해 보겠습니다.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4바이트 경계에 정렬된 단순 구조
   struct CharIntShort pack(4)
     {
      char              c; // sizeof(char)=1
      int               i; // sizeof(double)=4
      short             s; // sizeof(short)=2
     };
//--- 단순 구조 인스턴스 선언
   CharIntShort ch_in_sh;
//--- 각 구조 멤버의 크기 표시
   Print("sizeof(ch_in_sh.c)=",sizeof(ch_in_sh.c));
   Print("sizeof(ch_in_sh.i)=",sizeof(ch_in_sh.i));
   Print("sizeof(ch_in_sh.s)=",sizeof(ch_in_sh.s));
//--- POD 구조의 크기가 멤버의 크기의 합과 같은지 확인합니다.
   Print("sizeof(CharIntShort)=",sizeof(CharIntShort));
/*
  결과:
   sizeof(ch_in_sh.c)=1
   sizeof(ch_in_sh.i)=4
   sizeof(ch_in_sh.s)=2
   sizeof(CharIntShort)=12
```

© 2000-2025, MetaQuotes Ltd.

```
*/
  }
```

구조 내용은 변경되지 않았지만 멤버 시퀀스를 변경하면 크기가 증가했습니다.

상속할 때 정렬도 고려해야 합니다. 단일 'char' 유형 멤버가 있는 간단한 상위 구조를 사용하여 이를 시연해 보겠습니다. 정렬이 없는 구조 크기는 1입니다.

```
   struct Parent
     {
      char              c;     // sizeof(char)=1
     };
```

'short' (sizeof(short)=2) 유형 멤버를 특징으로 하는 Children child 클래스를 생성하겠습니다.

```
   struct Children pack(2) : Parent
     {
      short             s;    // sizeof(short)=2
     };
```

따라서 정렬을 2바이트로 설정할 때 구조 크기는 4와 같으나, 멤버 크기는 3입니다. 이 예에서는 자식 클래스의 'short' 필드에 대한 액세스가 2바이트로 정렬되도록 부모 클래스에 2바이트를 할당해야 합니다.

MQL5 응용 프로그램이 파일 또는 스트림 수준에서 쓰기/읽기를 통해 타사 데이터와 상호 작용하는 경우 구조 멤버에 대해 메모리를 할당하는 방법에 대한 지식이 필요합니다.

표준 라이브러리의 MQL5₩Include₩WinAPI 디렉토리에는 WinAPI 기능으로 작업하는 기능이 포함되어 있습니다. 이러한 기능은 WinAPI 작업에 필요한 경우에 대해 지정된 정렬을 가진 구조를 적용합니다.

offsetof는 pack 특성과 직접 관련된 특수 명령입니다. 이것은 구조 시작부터 멤버 오프셋을 얻을 수 있게 해줍니다.

```
//--- 자식 유형 변수 선언
   Children child;
//--- 구조의 시작부터 오프셋을 탐지합니다
   Print("offsetof(Children,c)=",offsetof(Children,c));
   Print("offsetof(Children,s)=",offsetof(Children,s));
/*
  결과:
   offsetof(Children,c)=0
   offsetof(Children,s)=2
*/
```

지정자 'final'

구조 선언 중에 'final' 지정자를 사용하면 이 구조로부터 추가 상속이 금지됩니다. 구조에 추가 수정이 필요하지 않거나 보안상의 이유로 수정이 허용되지 않는 경우 이 구조를 'final' 한정자로 선언하십시오. 또한, 구조의 모든 멤버는 암묵적으로 최종 멤버로 간주됩니다.

© 2000-2025, MetaQuotes Ltd.

```
struct settings final
  {
  //--- Structure body
  };

struct trade_settings : public settings
  {
  //--- 구조 바디
  };
```

위의 예와 같이 'final' 한정자를 사용하여 구조에서 상속하려고 하면 컴파일러에서 오류를 반환합니다:

```
'settings' 로부터 상속할 수 없습니다. 그것이 다음과 같이 선언되었으므로: 'final'
 'settings' 선언 참조
```

Classes

클래스는 다음에서 구조물과 다릅니다:

- 키워드 클래스는 선언에 사용됩니다;
- 기본적으로 모든 클래스 멤버는 달리 표시되지 않는 한 액세스 지정자를 비공개로 가집니다. 구조의 데이터-멤버는 달리 명시되지 않은 한 공개 접근의 기본 유형을 가집니다;
- 클래스 개체에는 클래스에 선언된 가상 함수가 없는 경우에도 항상 가상 함수 테이블이 있습니다. 구조에는 가상 함수를 사용할 수 없습니다;
- new 연산자를 클래스 객체에 적용할 수 있으며, 이 연산자는 구조물에 적용할 수 없습니다;
- 클래스는 클래스에서만 상속될 수 있으며, 구조는 구조에서만 상속될 수 있습니다.

클래스와 구조는 명시적 생성자 및 소멸자를 가질 수 있습니다. 생성자가 명시적으로 정의된 경우 초기화 시퀀스를 사용하여 구조 또는 클래스 변수를 초기화할 수 없습니다.

예제:

```
struct trade_settings
  {
   double take;          // 차익 확정 가격의 가치
   double stop;          // 보호 중지 가격의 가치
   uchar  slippage;      // 허용 저하 값
   //--- 구성자
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
   //--- 소멸자
         ~trade_settings() { Print("이것이 끝입니다"); }
  };
//--- 컴파일러에서 초기화가 불가능하다는 오류 메시지를 생성합니다
trade_settings my_set={0.0,0.0,5};
```

생성자 및 소멸자

생성자는 구조나 클래스의 개체를 만들 때 자동으로 호출되는 특수 함수이며 일반적으로 클래스 멤버를 초기화 하는 데 사용됩니다. 또한 달리 명시되지 않은 한 구조에도 동일하게 적용되는 반면, 클래스에만 대해 이야기하겠습니다. 생성자 이름은 클래스 이름과 일치해야 합니다. 생성자에 반환 유형이 없습니다 (void 유형을 지정할 수 있음).

© 2000-2025, MetaQuotes Ltd.

정의된 클래스 멤버 – 문자열, 동적 배열 및 초기화가 필요한 개체 –는 생성자가 있는지 여부에 관계없이 초기화됩니다.

각 클래스에는 매개 변수 수와 초기화 목록에 따라 다른 생성자가 여러 개 있을 수 있습니다. 매개변수를 지정해야 하는 생성자를 모수 생성자라고 합니다.

매개 변수가 없는 생성자를 **기본 생성자**라고 합니다. 클래스에 선언된 생성자가 없는 경우 컴파일러는 컴파일 중에 기본 생성자를 만듭니다.

```
//+------------------------------------------------------------------+
//| 날짜 작업을 위한 클래스                                          |
//+------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;          // 연도
   int               m_month;         // 월
   int               m_day;           // 일
   int               m_hour;          // 시
   int               m_minute;        // 분
   int               m_second;        // 초
public:
   //--- 기본 생성자
                     MyDateClass(void);
   //--- 매개변수 생성자
                     MyDateClass(int h,int m,int s);
  };
```

생성자를 클래스 설명에 선언한 다음 해당 본문을 정의할 수 있습니다. 예를 들어 MyDateClass의 두 생성자는 다음과 같이 정의할 수 있습니다:

```
//+------------------------------------------------------------------+
//| 기본 생성자                                                      |
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(void)
  {
//---
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=mdt.hour;
   m_minute=mdt.min;
   m_second=mdt.sec;
   Print(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| 매개변수 생성자                                                  |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
MyDateClass::MyDateClass(int h,int m,int s)
  {
   MqlDateTime mdt;
   datetime t=TimeCurrent(mdt);
   m_year=mdt.year;
   m_month=mdt.mon;
   m_day=mdt.day;
   m_hour=h;
   m_minute=m;
   m_second=s;
   Print(__FUNCTION__);
  }
```

기본 생성자에서 클래스의 모든 멤버는 TimeCurrent() 함수를 사용하여 채워집니다. 매개변수 생성자에서는 시간 값만 입력됩니다. 클래스의 다른 멤버(m_year, m_month and m_day)은 현재 날짜에 자동으로 초기화됩니다.

기본 생성자는 해당 클래스의 객체 배열을 초기화할 때 특별한 목적을 가집니다. 기본값이 있는 모든 매개 변수 생성자는 기본 생성자가 **아닙니다**. 여기 예제가 있습니다:

```
//+------------------------------------------------------------------+
//| 기본 생성자가 있는 클래스                                          |
//+------------------------------------------------------------------+
class CFoo
  {
   datetime          m_call_time;     // 마지막 객체 호출 시간
public:
   //--- 기본값이 있는 매개 변수를 가진 생성자가 기본 생성자가 아닙니다
                     CFoo(const datetime t=0){m_call_time=t;};
   //--- 복제 생성자
                     CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};

   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
// CFoo foo; // 이 변수를 사용할 수 없습니다. 기본 생성자가 설정되지 않았습니다.
//--- CFoo 객체를 만드는 데 사용할 수 있는 옵션
   CFoo foo1(TimeCurrent());     // 파라메트릭 생성자의 명시적 호출
   CFoo foo2();                  // 기본 파라미터가 있는 모수 생성자의 명시적 호출
   CFoo foo3=D'2009.09.09';      // 모수 생성자의 암시적 호출
   CFoo foo40(foo1);             // 복제 생성자의 명시적 호출
   CFoo foo41=foo1;              // 복제 생성자의 암시적 호출
   CFoo foo5;                    // 기본 생성자의 명시적 호출(기본 생성자가 없는 경우,
                                 // 그러면 기본값인 모수 생성자를 호출합니다)
//--- CFoo 포인터를 수신할 수 있는 옵션
```

© 2000-2025, MetaQuotes Ltd.

```
   CFoo *pfoo6=new CFoo();        // 객체의 동적 생성 및 객체에 대한 포인터
   CFoo *pfoo7=new CFoo(TimeCurrent());// 동적 객체 생성의 또 다른 옵션
   CFoo *pfoo8=GetPointer(foo1); // 이제 pfoo8 포인트와 객체 foo1 연결
   CFoo *pfoo9=pfoo7;            // pfoo9 및 pfoo7이 동일한 개체를 가리킵니다.
   // CFoo foo_array[3];         // 이 옵션을 사용할 수 없습니다 - 기본 생성자가 지정되지 않
//--- m_call_time 값 보이기
   Print("foo1.m_call_time=",foo1.ToString());
   Print("foo2.m_call_time=",foo2.ToString());
   Print("foo3.m_call_time=",foo3.ToString());
   Print("foo4.m_call_time=",foo4.ToString());
   Print("foo5.m_call_time=",foo5.ToString());
   Print("pfoo6.m_call_time=",pfoo6.ToString());
   Print("pfoo7.m_call_time=",pfoo7.ToString());
   Print("pfoo8.m_call_time=",pfoo8.ToString());
   Print("pfoo9.m_call_time=",pfoo9.ToString());
//--- 동적으로 작성된 배열 삭제
   delete pfoo6;
   delete pfoo7;
   //pfoo8 삭제;  // pfoo8은 자동으로 생성된 개체 foo1을 가리키므로 명시적으로 삭제할 필요가 없
   //pfoo9 삭제;  // pfoo9은 명시적으로 삭제할 필요가 없습니다. pfoo7과 같은 객체를 가리키기 때
  }
```

이 문자열에 대한 설명을 해제하면

```
  //CFoo foo_array[3];     // 이 변수를 사용할 수 없습니다. 기본 생성자가 설정되지 않았습니다
```

또는

```
  //CFoo foo_dyn_array[];  // 이 변수를 사용할 수 없습니다 - 기본 생성자가 설정되지 않았습니다
```

그러면 컴파일러는 "기본 생성자가 정의되지 않음"에 대한 오류를 반환합니다.

클래스에 커스텀 생성자가 있는 경우 컴파일러에서 기본 생성자를 생성하지 않습니다. 즉, 모수 생성자가 클래스에 선언되었지만 기본 생성자는 선언되지 않은 경우 이 클래스의 개체 배열을 선언할 수 없습니다. 컴파일러에서 이 스크립트에 대한 오류를 반환합니다:

```
//+------------------------------------------------------------------+
//| 기본 생성자가 없는 클래스                                        |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 컴파일 중 "기본 생성자가 정의되지 않았습니다" 오류를 가져옵니다
```

© 2000-2025, MetaQuotes Ltd.

```
   CFoo badFoo[5];
  }
```

이 예제에서 CFoo 클래스에는 선언된 파라미터 생성자가 있습니다 - 이 경우 컴파일러는 컴파일 중에 기본 생성자를 자동으로 만들지 않습니다. 개체 배열을 선언하는 동시에 모든 개체를 자동으로 생성 및 초기화해야 한다고 가정합니다. 개체의 자동 초기화 중에는 기본 생성자를 호출해야 하지만 기본 생성자는 명시적으로 선언되지 않고 컴파일러에 의해 자동으로 생성되지 않으므로 이러한 개체를 만들 수 없습니다. 이러한 이유로 컴파일러는 컴파일 단계에서 오류를 생성합니다.

생성자를 사용하여 개체를 초기화하는 특수 구문이 있습니다. 구조체 또는 클래스 멤버에 대한 생성자 이니셜라이저(초기화를 위한 특수 구성)를 초기화 목록에 지정할 수 있습니다.

초기화 목록은 쉼표로 구분된 이니셜라이저 목록으로, 생성자의 매개 변수 목록 뒤 콜론 뒤에 오고 본문(열림 꺾쇠 앞에 있음) 앞에 나옵니다. 몇 가지 요구 사항이 있습니다:

- 초기화 목록은 생성자에서만 사용할 수 있습니다;
- 부모 멤버는 초기화 목록에서 초기화할 수 없습니다;
- 초기화 목록 뒤에는 함수의 정의(실행)가 와야 합니다.

다음은 클래스 멤버를 초기화하기 위한 여러 생성자의 예입니다.

```
//+------------------------------------------------------------------+
//| 다음은 클래스 멤버를 초기화하기 위한 여러 생성자의 예입니다                    |
//+------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // 첫째 이름
   string            m_second_name;    // 둘째 이름
public:
   //--- 비어 있는 기본 생성자
                     CPerson() {Print(__FUNCTION__);};
   //--- 모수 생성자
                     CPerson(string full_name);
   //--- 초기화 목록이 있는 생성자
                     CPerson(string surname,string name): m_second_name(surname), m_fi
   void PrintName(){PrintFormat("Name=%s Surname=%s",m_first_name,m_second_name);};
  };
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
CPerson::CPerson(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>=0)
     {
      m_first_name=StringSubstr(full_name,0,pos);
      m_second_name=StringSubstr(full_name,pos+1);
     }
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                          |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- "기본 생성자가 정의되지 않았습니다" 오류가 표시됩니다
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                          // 톰 소여
   CPerson Huck("Huckleberry","Finn");                // 허클베리 핀
   CPerson *Pooh = new CPerson("Winnie","Pooh");  // 곰돌이 푸
   //--- 결과 값
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //--- 동적으로 작성된 객체 삭제
   delete Pooh;
  }
```

이 경우 CPerson 클래스에는 세 개의 생성자가 있습니다:

1. 이 클래스의 객체 배열을 작성할 수 있는 명시적인 기본 생성자;
2. 매개 변수가 하나 있는 생성자로, 매개 변수로 전체 이름을 가져오고 찾은 공간에 따라 이름과 두 번째 이름으로 나눕니다;
3. 초기화 목록을 포함하는 두 개의 매개 변수를 가진 생성자. 이니셜라이저 - m_second_name(성) 및 m_first_name(이름).

목록을 사용한 초기화가 할당을 대체했습니다. 개별 멤버는 다음과 같이 초기화해야 합니다:

```
class_member (표현식 목록)
```

초기화 리스트에서 멤버들은 어떤 순서로든 갈 수 있지만, 모든 클래스 멤버들은 발표 순서에 따라 초기화됩니다. 즉, 세 번째 생성자에서 먼저 m_first_name 멤버가 먼저 발표되는 대로 초기화되며, m_second_name 이 초기화되어야 합니다. 이는 일부 클래스 멤버의 초기화가 다른 클래스 멤버의 가치에 따라 달라지는 경우에 고려되어야 한다.

기본 생성자가 기본 클래스에 선언되지 않은 동시에 매개 변수를 가진 하나 이상의 생성자가 선언된 경우 초기화 목록에서 항상 기본 클래스 생성자 중 하나를 호출해야 합니다. 쉼표를 통해 목록의 일반 멤버로 사용되며, 초기화 목록의 어디에 있든 개체를 초기화하는 동안 먼저 호출됩니다.

```
//+------------------------------------------------------------------+
//| 기초 클래스                                                      |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //--- 초기화 목록이 있는 생성자
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------+
//| CFoo에서 파생된 클래스                                           |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
class CBar : CFoo
  {
   CFoo              m_member;      // 클래스 멤버는 상위의 개체입니다
public:
   //--- 초기화 목록의 기본 생성자는 상위 생성자를 호출합니다
                     CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   CBar bar;
  }
```

이 예에서는 막대 객체를 만들 때 기본 생성자 CBar()가 호출됩니다. 여기서 먼저 상위 CFoo에 대한 생성자를 호출한 다음 m_member 클래스 멤버에 대한 생성자를 호출합니다.

소멸자는 클래스 개체가 제거될 때 자동으로 호출되는 특수 함수입니다. 소멸자의 이름은 물결표(~)와 함께 클래스 이름으로 쓰여 있습니다. 초기화 취소가 필요한 문자열, 동적 배열 및 개체는 소멸자의 유무에 관계없이 초기화 취소됩니다. 소멸자가 있는 경우 이러한 작업은 소멸자를 호출한 후 수행됩니다.

소멸자는 가상 키워드로 선언되었는지 여부와 관계없이 항상 가상입니다.

클래스 메서드 정의

클래스 함수 메서드는 클래스 선언 내부와 외부에서 모두 정의할 수 있습니다. 메서드가 클래스 내에 정의되어 있으면 메서드 선언 바로 뒤에 해당 본문이 나옵니다.

예제:

```
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
```

© 2000-2025, MetaQuotes Ltd.

```
   int               GetXSize()                  { return(m_xsize);        }
   int               GetType()                   { return(m_type);         }
   void              Left()                      { m_xpos-=SHAPE_SIZE;     }
   void              Right()                     { m_xpos+=SHAPE_SIZE;     }
   void              Rotate()                    { m_prev_turn=m_turn; if(++m_turn>3) r
   virtual void      Draw()                      { return;                 }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
  };
```

SetRightBorder(int 보더)에서 Draw()까지의 함수는 CTetrisShape 클래스 내에서 직접 선언 및 정의됩니다.

CTetrisShape() 생성자 및 메서드 CheckDown(int& pad_array[]), CheckLeft(int& side_row[]) 및 CheckRight(int& side_row[])는 클래스 내에서만 선언되며 아직 정의되지 않았습니다. 이러한 함수의 정의는 코드에 자세히 나와 있습니다. 클래스 외부에서 메서드를 정의하기 위해 범위 지정 연산자가 사용되고 클래스 이름이 스코프로 사용됩니다.

예제:

```
//+------------------------------------------------------------------+
//| 기본 클래스 생성자                                               |
//+------------------------------------------------------------------+
void CTetrisShape::CTetrisShape()
  {
   m_type=0;
   m_ypos=0;
   m_xpos=0;
   m_xsize=SHAPE_SIZE;
   m_ysize=SHAPE_SIZE;
   m_prev_turn=0;
   m_turn=0;
   m_right_border=0;
  }
//+------------------------------------------------------------------+
//| 아래로 이동하는 기능 확인(스틱 및 큐브)                          |
//+------------------------------------------------------------------+
bool CTetrisShape::CheckDown(int& pad_array[])
  {
   int i,xsize=m_xsize/SHAPE_SIZE;
//---
   for(i=0; i<xsize; i++)
     {
      if(m_ypos+m_ysize>=pad_array[i]) return(false);
     }
//---
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

퍼블릭, 보호 및 프라이빗 액세스 지정자

새로운 클래스를 개발할 때는 외부로부터의 멤버 접근을 제한하는 것이 좋습니다. 이러한 목적 탓에 프라이빗 또는 보호가 사용됩니다. 이 경우 숨겨진 데이터는 동일한 클래스의 함수 방식에서만 액세스할 수 있습니다. *protected* 키워드를 사용하는 경우, 클레스 메서드 – 클래스 상속 –에서도 숨겨진 데이터에 엑세스할 수 있습니다. 동일한 방법을 사용하여 클래스의 함수 메서드에 대한 액세스를 제한할 수 있습니다 .

클래스의 멤버 및/또는 메서드에 대한 액세스를 완전히 열어야 하는 경우 키워드 public을 사용하십시오.

예제:

```
class CTetrisField
  {
private:
   int               m_score;                              // 스코어
   int               m_ypos;                               // 그림의 현재 위치
   int               m_field[FIELD_HEIGHT][FIELD_WIDTH]; // 우물 행렬
   int               m_rows[FIELD_HEIGHT];                 // 우물 행 번호 지정
   int               m_last_row;                           // 마지막 자유 행
   CTetrisShape     *m_shape;                              // 테트리스 모형
   bool              m_bover;                              // 게임 오버
public:
   void              CTetrisField() { m_shape=NULL; m_bover=false; }
   void              Init();
   void              Deinit();
   void              Down();
   void              Left();
   void              Right();
   void              Rotate();
   void              Drop();
private:
   void              NewShape();
   void              CheckAndDeleteRows();
   void              LabelOver();
  };
```

지정자 **public:** (다음 액세스 지정자 이전) 뒤에 선언된 클래스 멤버 및 메서드는 프로그램에 의한 클래스 개체에 대한 참조에서 사용할 수 있습니다. 이 예에서는 CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() 및 Drop() 함수가 사용됩니다.

private: 요소의 액세스 지정자 다음에 선언된 멤버(다음 액세스 지정자 이전)는 이 클래스의 멤버 함수에서만 사용할 수 있습니다. 요소에 대한 액세스 지정자는 항상 콜론(:)으로 끝나며 클래스 정의에 여러 번 나타날 수 있습니다.

protected: 액세스 지정자 뒤에 선언된 모든 클래스 멤버(및 다음 액세스 지정자까지)는 이 클래스의 멤버-함수와 클래스 자식 항목의 멤버-함수에만 사용할 수 있습니다. 외부에서 프라이빗 및 보호 지정자가 있는 멤버를 참조하려고 하면 컴파일 스테이지 오류가 발생합니다. 예제:

```
class A
  {
```

© 2000-2025, MetaQuotes Ltd.

```
protected:
   //--- 복제 연산자는 클래스 A와 그 자식 항목에서만 사용할 수 있습니다.
   void operator=(const A &)
     {
     }
  };
class B
  {
   //--- 선언된 클래스 A 객체
   A                 a;
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- B형 변수 두 개를 선언
   B b1, b2;
   //--- 어떤 객체를 다른 객체로 복사 시도
   b2=b1;
  }
```

코드를 컴파일할 때 원격 복사 연산자를 호출하려는 시도와 같은 오류 메시지가 수신됩니다:

```
attempting to reference deleted function 'void B::operator=(const B&)'   trash3.mq5
```

아래의 두 번째 문자열은 클래스 A의 사용할 수 없는 복제 연산자를 호출하기 때문에 클래스 B의 복제 연산자가 명시적으로 삭제되었습니다.

```
   function 'void B::operator=(const B&)' was implicitly deleted because it invokes in
```

기본 클래스의 멤버에 대한 액세스는 파생 클래스에서 상속 중에 재정의할 수 있습니다.

'삭제' 지정자

삭제 지정자는 사용할 수 없는 클래스 멤버를 표시합니다. 즉, 프로그램이 명시적 또는 암시적으로 이러한 함수를 참조하는 경우 컴파일 단계에서 오류가 이미 수신됩니다. 예를 들어 이 지정자를 사용하면 자식 클래스에서 부모 메서드를 사용할 수 없게 만들 수 있습니다. 상위 클래스의 프라이빗 영역 (프라이빗 섹션으로의 선언)에 함수를 선언해도 동일한 결과를 얻을 수 있습니다. 여기서 삭제를 사용하면 하위 항목 수준에서 코드를 더 잘 읽고 관리할 수 있습니다.

```
class A
  {
public:
                     A(void) {value=5;};
   double            GetValue(void) {return(value);}
private:
   double            value;
  };
class B: public A
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   double            GetValue(void)=delete;
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- A형 변수 선언
   A a;
   Print("a.GetValue()=", a.GetValue());
//--- B 유형 변수에서 값을 가져오려 시도
   B b;
   Print("b.GetValue()=", b.GetValue()); // 컴파일러에서 이 문자열에 오류를 표시합니다
  }
```

컴파일러 메시지:

```
attempting to reference deleted function 'double B::GetValue()'
   function 'double B::GetValue()' was explicitly deleted here
```

'삭제' 지정자는 자동 캐스팅 또는 복사 생성자를 비활성화할 수 있습니다. 그렇지 않으면 프라이빗 섹션에도 숨겨야 합니다. 예제:

```
class A
  {
public:
   void              SetValue(double v) {value=v;}
   //--- int 타입 호출 비활성화
   void              SetValue(int) = delete;
   //--- 복제 연산자 비활성화
   void              operator=(const A&) = delete;
private:
   double            value;
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- A형 변수 두 개 선언
   A a1, a2;
   a1.SetValue(3);       // 오류!
   a1.SetValue(3.14);    // OK
   a2=a1;                // 오류!
  }
```

컴파일하는 동안 오류 메시지가 표시됩니다:

```
attempting to reference deleted function 'void A::SetValue(int)'
   function 'void A::SetValue(int)' was explicitly deleted here
attempting to reference deleted function 'void A::operator=(const A&)'
```

© 2000-2025, MetaQuotes Ltd.

```
   function 'void A::operator=(const A&)' was explicitly deleted here
```

지정자 'final'

클래스 선언 중에 'final' 지정자를 사용하면 이 클래스에서 더 이상 상속할 수 없습니다. 클래스 인터페이스에 추가 수정이 필요하지 않거나 보안상의 이유로 수정이 허용되지 않는 경우 이 클래스를 'final' 한정자로 선언하십시오. 또한 클래스의 모든 멤버들은 암묵적으로 final로 간주됩니다.

```
class CFoo final
  {
  //--- Class body
  };

class CBar : public CFoo
  {
  //--- Class body
  };
```

위의 예와 같이 'final' 지정자가 있는 폼 클래스를 상속하려고 하면 컴파일러에서 오류를 반환합니다:

```
'CFoo'가 'final'로 선언되었으므로 'CFoo'에서 상속할 수 없습니다
'CFoo' 선언 참조
```

Unions (union)

Union은 동일한 메모리 영역을 공유하는 여러 변수로 구성된 특수 데이터 유형입니다. 따라서 유니언은 동일한 비트 시퀀스를 두 가지(또는 그 이상) 다른 방법으로 해석할 수 있는 기능을 제공합니다. 유니언 선언은 구조 선언과 유사하며 union 키워드로 시작합니다.

```
union LongDouble
{
  long   long_value;
  double double_value;
};
```

구조와는 달리 다양한 유니언 멤버들이 같은 메모리 영역에 속해 있습니다. 이 예에서 LongDouble의 결합은 동일한 메모리 영역을 공유하는 long 및 double 유형 값으로 선언됩니다. long_value 변수와 double_value 변수가 (메모리에) 겹치기 때문에 유니언 저장소는 *long* 정수 값과 *double* 실수 값을 동시에 저장할 수 없습니다. 반면 MQL5 프로그램은 결합에 포함된 데이터를 언제든지 정수(long) 또는 실수(double) 값으로 처리할 수 있습니다. 따라서 통합에서는 동일한 데이터 시퀀스를 나타내기 위해 두 개 이상의 옵션을 수신할 수 있습니다.

유니언 선언 중에 컴파일러는 변수 유니언에 largest 타입(볼륨 기준)을 저장하기에 충분한 메모리 영역을 자동으로 할당합니다. 구조 -포인트 연산자와 동일한 구문이 유니언 요소에 액세스하는 데 사용됩니다.

```
union LongDouble
{
  long   long_value;
  double double_value;
};
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//---
   LongDouble lb;
//--- 잘못된 -nan(ind) 수를 가져와 표시
   lb.double_value=MathArcsin(2.0);
   printf("1.  double=%f                integer=%I64X",lb.double_value,lb.long_value);
//--- 최대 정규화 값 (DBL_MAX)
   lb.long_value=0x7FEFFFFFFFFFFFFF;
   printf("2.  double=%.16e  integer=%I64X",lb.double_value,lb.long_value);
//--- 최소 양의 정규화 (DBL_MIN)
   lb.long_value=0x0010000000000000;
   printf("3.  double=%.16e  integer=%.16I64X",lb.double_value,lb.long_value);
  }
/*  실행 결과
    1.  double=-nan(ind)                integer=FFF8000000000000
    2.  double=1.7976931348623157e+308  integer=7FEFFFFFFFFFFFFF
    3.  double=2.2250738585072014e-308  integer=0010000000000000
*/
```

유니언은 프로그램이 동일한 메모리 데이터를 다른 방식으로 해석할 수 있도록 하기 때문에 비정상적인 유형 변환이 필요할 때 종종 사용됩니다.

유니언이 상속에 관여할 수도 없고, 특성상 정적 멤버를 둘 수도 없습니다. 다른 모든 측면에서, *유니언*은 모든 멤버들이 0의 오프셋을 갖는 구조처럼 행동합니다. 다음 유형은 유니언 멤버일 수 없습니다:

- 동적 배열
- 문자열
- 포인터 -오브젝트에 대한- 및 함수
- 클래스 오브젝트
- 구조 객체에 생성자 또는 소멸자가 있음
- 구조 객체: 1-5 포인트에 멤버가 있음

클래스와 비슷하게, 유니언은 메소드뿐만 아니라 생성자와 소멸자를 가질 수 있습니다. 기본적으로, 유니언은 퍼블릭 액세스 유형입니다. 프라이빗 요소를 생성하려면, 프라이빗 키워드를 사용하십시오. color 유형의 색상을 ARGB로 변환하는 방법을 보여주는 예제에 이러한 모든 가능성이 표시됩니다. ColorToARGB() 함수와 마찬가지로.

```
//+------------------------------------------------------------------+
//| color(BGR) 유니언의 ARGB로의 변환                                |
//+------------------------------------------------------------------+
union ARGB
  {
   uchar             argb[4];
   color             clr;
   //--- 생성자
                     ARGB(color col,uchar a=0){Color(col,a);};
                    ~ARGB(){};
   //--- 퍼블릭 메서드
public:
```

© 2000-2025, MetaQuotes Ltd.

```
   uchar   Alpha(){return(argb[3]);};
   void    Alpha(const uchar alpha){argb[3]=alpha;};
   color   Color(){ return(color(clr));};
   //--- 프라이빗 메서드
private:
   //+------------------------------------------------------------------+
   //| 알파 채널 값 및 색상 설정                                        |
   //+------------------------------------------------------------------+
   void    Color(color col,uchar alpha)
     {
      //--- clr 멤버로 색상 설정
      clr=col;
      //--- 알파 성분 값 설정 - 불투명도 수준
      argb[3]=alpha;
      //--- R 및 B 구성 요소의 바이트 교환(빨간색 및 파란색)
      uchar t=argb[0];argb[0]=argb[2];argb[2]=t;
     };
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 0x55는 55/255=21.6%(0% - 완전 투명)를 의미합니다
   uchar alpha=0x55;
//--- 색상 유형은 0x00BBGGRR 처럼 표현됩니다
   color test_color=clrDarkOrange;
//--- ARGB 유니언의 바이트 값이 여기에 허용됩니다
   uchar argb[];
   PrintFormat("0x%8X - %s 'color' 유형은 다음과 같습니다, BGR=(%s)",
               test_color,ColorToString(test_color,true),ColorToString(test_color));
//--- ARGB 유형은 0x00RRGGBB 같이 표시되며, RR 및 BB 구성요소는 스왑됩니다
   ARGB argb_color(test_color);
//--- 바이트 배열을 복제
   ArrayCopy(argb,argb_color.argb);
//--- 다음은 ARGB 표현의 모습입니다
   PrintFormat("0x%8X - ARGB 표현, 알파 채널=0x%2x, ARGB=(%d,%d,%d,%d)",
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- 불투명도 수준 추가
   argb_color.Alpha(alpha);
//--- ARGB를 'color' 유형으로 정의 시도
   Print("ARGB as color=(",argb_color.clr,")  alpha channel=",argb_color.Alpha());
//--- 바이트 배열을 복제
   ArrayCopy(argb,argb_color.argb);
//--- 다음은 ARGB 표현의 모습입니다
   PrintFormat("0x%8X - ARGB 표현, 알파 채널=0x%2x, ARGB=(%d,%d,%d,%d)",
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- ColorToARGB() 함수 결과를 확인
   PrintFormat("0x%8X - ColorToARGB(%s,0x%2x) 결과",ColorToARGB(test_color,alpha),
```

© 2000-2025, MetaQuotes Ltd.

```
                ColorToString(test_color,true),alpha);
  }
/* 실행 결과
   0x00008CFF - clrDarkOrange, BGR=(255,140,0)의 color 유형은 이것과 같습니다
  0x00FF8C00 - ARGB 표현, 알파 채널=0x00, ARGB=(0,255,140,0)
   ARGB as color=(0,140,255)  alpha channel=85
   0x55FF8C00 - ARGB 표현, 알파 채널=0x55, ARGB=(85,255,140,0)
   0x55FF8C00 - ColorToARGB(clrDarkOrange,0x55)의 결과
*/
```

인터페이스

인터페이스는 클래스에서 구현할 수 있는 특정 기능을 결정할 수 있도록 합니다. 실제로 인터페이스는 멤버를 포함할 수 없으며 생성자 및/또는 소멸자가 없을 수 있는 클래스입니다. 인터페이스에 선언된 모든 메서드는 명시적 정의가 없어도 순수하게 가상입니다.

인터페이스는 "interface" 키워드를 사용하여 정의됩니다. 예제:

```
//--- 동물 설명을 위한 기본 인터페이스
interface IAnimal
  {
//--- 인터페이스의 메서드는 기본적으로 공용 액세스 권한을 가집니다
   void Sound();  // 동물이 내는 소리
  };
//+------------------------------------------------------------------+
//|  CCat 클래스는 IAnimal 인터페이스에서 상속됩니다                 |
//+------------------------------------------------------------------+
class CCat : public IAnimal
  {
public:
                     CCat() { Print("고양이가 태어났습니다"); }
                    ~CCat() { Print("고양이가 죽었습니다");  }
   //--- IAnimal 인터페이스의 사운드 메서드 구현
   void Sound(){ Print("야옹"); }
  };
//+------------------------------------------------------------------+
//|  CDog 클래스는 IAnimal 인터페이스에서 상속됩니다                 |
//+------------------------------------------------------------------+
class CDog : public IAnimal
  {
public:
                     CDog() { Print("개가 태어났습니다"); }
                    ~CDog() { Print("개가 죽었습니다");  }
   //--- IAnimal 인터페이스의 사운드 메서드 구현
   void Sound(){ Print("멍멍"); }
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- IAnimal 유형 개체에 대한 포인터 배열
   IAnimal *animals[2];
//--- IAnimal의 하위 클래스를 만들고 해당 클래스에 대한 포인터를 배열에 저장합니다
   animals[0]=new CCat;
   animals[1]=new CDog;
//--- 각 자식 항목에 대한 기본 IAnimal 인터페이스의 Sound() 메서드 호출
   for(int i=0;i<ArraySize(animals);++i)
      animals[i].Sound();
//--- 객체 삭제
   for(int i=0;i<ArraySize(animals);++i)
      delete animals[i];
//--- 실행 결과
/*
  고양이가 태어났습니다
  개가 태어났습니다
  야옹
  멍멍
  고양이가 죽었습니다
  개가 죽었습니다
*/
  }
```

추상 클래스와 마찬가지로 인터페이스 객체는 상속 없이 만들 수 없습니다. 인터페이스는 다른 인터페이스에서만 상속할 수 있으며 클래스의 부모 인터페이스일 수 있습니다. 인터페이스는 항상 공개적으로 가시적입니다.

클래스 또는 구조 선언 내에 인터페이스를 선언할 수 없지만 인터페이스에 대한 포인터는 void * 유형의 변수에 저장할 수 있습니다. 일반적으로 클래스의 개체에 대한 포인터는 void * 유형의 변수에 저장할 수 있습니다. void * 포인터를 특정 클래스의 객체로 변환하려면 dynamic_cast 연산자를 사용하십시오.변환할 수 없는 경우 dynamic_cast 작업의 결과는 NULL이 됩니다.

더 보기

객체지향 프로그래밍

© 2000-2025, MetaQuotes Ltd.

동적 배열 객체

동적 배열

최대 4차원 배열을 선언할 수 있습니다. 동적 배열(첫 번째 대괄호 쌍에 지정되지 않은 값의 배열)을 선언하면 컴파일러는 자동으로 위 구조의 변수(동적 배열 개체)를 만들고 올바른 초기화를 위한 코드를 제공합니다.

동적 어레이는 선언된 블록의 가시성 영역을 벗어나면 자동으로 해제됩니다.

예제:

```
double matrix[][10][20]; // 3차원 동적 배열
ArrayResize(matrix,5);   // 첫번째 차원의 크기 설정
```

정적 배열

모든 중요한 배열 치수가 명시적으로 지정되면 컴파일러는 필요한 메모리 크기를 미리 할당합니다. 이러한 배열을 정적 배열이라고 합니다. 그럼에도 불구하고 컴파일러는 동적 배열의 개체에 추가 메모리를 할당합니다. 동적 배열은 (개체) 사전 할당된 정적 버퍼(배열을 저장하기 위한 메모리 부분)와 연결됩니다.

동적 배열 개체를 만드는 이유는 이 정적 배열을 일부 함수에 매개 변수로 전달해야 하기 때문입니다.

예제:

```
double stat_array[5]; // 1차원 정적 배열
some_function(stat_array);
...
bool some_function(double& array[])
  {
   if(ArrayResize(array,100)<0) return(false);
   ...
   return(true);
  }
```

구조의 배열

정적 배열이 구조의 멤버로 선언될 때 동적 배열 개체는 생성되지 않습니다. 이는 Windows API에서 사용되는 데이터 구조의 호환성을 보장하기 위해 수행됩니다.

그러나 구조물의 멤버로 선언된 정적 어레이는 MQL5 함수에 전달될 수도 있습니다. 이 경우 매개 변수를 전달하면 동적 배열의 임시 개체가 생성됩니다. 이러한 개체는 정적 배열 -구조의 멤버와 연결됩니다.

더 보기

배열 함수, 변수 초기화, 가시성 범위 및 변수 수명, 객체 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

매트릭스와 벡터

Type벡터는 벡터 작업을 가능하게 하는 MQL5의 특수 데이터 유형입니다. 벡터는 double 타입의 1차원 배열입니다. 벡터는 물리학, 기하학 등 많은 과학 분야에서 사용되는 선형 대수의 기본 개념 중 하나입니다. 벡터는 3D 그래픽 및 기타 응용 분야에서 선형 방정식 시스템을 푸는 데 사용됩니다. 벡터는 더해지고 곱해질 수 있습니다. 벡터 사이의 길이 또는 거리는 Norm을 통해 얻을 수 있습니다. 프로그래밍에서 벡터는 일반적으로 정규 벡터 연산이 없는 동종 요소의 배열로 표현됩니다 즉, 배열을 더하거나 곱할 수 없는 경우, norm이 없는 경우.

벡터는 행렬로 작업할 때 행 벡터와 문자열 벡터로 나타낼 수 있습니다. 또한 선형 대수학의 벡터는 공분산과 반공분산의 개념을 사용합니다. 이러한 개념은 MQL5 코드를 작성할 때 아무런 차이가 없습니다. 프로그래머만 벡터 유형의 각 객체가 무엇인지 결정하기 때문입니다. 예를 들어, 3D 그래픽의 회전, 변위 또는 압축 벡터의 경우입니다.

일반적으로 선형 대수학의 관점에서 볼 때 숫자도 벡터이지만 1차원 벡터 공간에 있습니다. 벡터 자체는 행렬의 특수한 경우로 간주할 수 있습니다.

Ypematrix는 행렬을 나타내는 MQL5의 또 다른 특수 데이터 유형입니다. 행렬은 실제로 double 유형의 2차원 배열입니다. 특정 유형의 데이터 세트에 대한 더 쉬운 작업을 위해 벡터와 행렬이 MQL5에 도입되었습니다. 이를 통해 개발자는 단순하고 수학과 같은 형태로 선형 대수의 이점을 얻을 수 있습니다. Matrix는 선형 또는 미분 방정식 시스템을 간결하게 작성하기 위해 사용할 수 있습니다. 행렬 행의 수는 방정식의 수에 해당하고 열의 수는 미지수와 같습니다. 결과적으로 선형 방정식은 행렬 연산을 통해 풀 수 있습니다.

행렬에 대해 다음 대수 연산이 정의됩니다:

- 동일한 크기의 행렬 추가
- 적합한 크기 행렬의 곱셈: 왼쪽 행렬의 열 개수는 오른쪽 행렬의 행 개수와 같아야 합니다.
- 열 벡터로 행렬 곱하기 행렬 곱셈 규칙에 따라 행 벡터를 행렬로 곱합니다. 이러한 관점에서 벡터는 행렬의 특수한 경우입니다.
- 숫자, 즉 스칼라로 행렬 곱하기

수학은 다양한 행렬 유형을 고려합니다. 예를 들어 단위 행렬, 대칭, 비대칭 대칭, 상부 및 하부 삼각 행렬 및 기타 유형이 있습니다. 다양한 정규형은 행렬 이론에서 중요한 역할을 합니다. 그것들은 특정 변환을 통해 얻을 수 있는 행렬의 특정 표준 형식을 나타냅니다. 실제로는 안정성과 같은 추가 속성이 있는 일반 형식이 사용됩니다.

벡터와 행렬을 사용하거나 관련 유형의 특수 메서드를 사용하면 수학 표기법에 가까운 더 간단하면서 간결하고 명확한 코드를 생성할 수 있습니다. 이러한 방법을 사용하면 중첩 루프를 생성하거나 계산시 배열의 올바른 인덱싱을 염두에 둘 필요가 없습니다. 따라서 이러한 방법을 사용하면 복잡한 프로그램을 개발할 때 안정성과 속도가 향상됩니다.

행렬 및 벡터 메서드 목록

Typesmatrix 그리고벡터는 넘파이 라이브러리 메서드에 해당하는 메서드를 포함합니다. 이러한 방법을 사용하면 최소한의 노력으로 알고리즘과 코드를 Python에서 MQL5로 변환할 수 있습니다. 많은 데이터 처리 작업, 수학 방정식, 신경망 및 기계 학습 작업은 이미 만들어진 Python 메서드와 라이브러리를 사용하여 해결할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

메서드 matrix/vector	NumPy의 Analogous 메서드	설명
void matrix.Eye(const int rows, const int cols, const int ndiag=0)	eye	대각선에 1이 있고 다른 곳에 0이 있는 행렬 구성
void matrix.Identity(const int rows)	identity	주대각선에 1이 있는 정사각 행렬을 구성합니다.
void matrix.Ones(const int rows, const int cols)	ones	1로 채워진 주어진 행과 열의 새로운 행렬을 구성합니다.
void matrix.Zeros(const int rows, const int cols)	zeros	0으로 채워진 주어진 행과 열로 새로운 행렬을 구성합니다.
void matrix.Full(const int rows, const int cols, const scalar value)	full	스칼라 값으로 채워진 주어진 행과 열의 새 행렬을 구성합니다.
void matrix.Copy(const matrix a)	copy	주어진 행렬의 복사본을 구성합니다.
void matrix.FromBuffer(const int rows, const int cols, const scalar array[], const int count=-1, const int offset=0)	frombuffer	1차원 배열에서 생성된 행렬 생성
void matrix.FromFile(const int rows, condt int cols, const int file_handle, const int count=-1, const int offset=0)	fromfile	텍스트 또는 이진 파일의 데이터에서 행렬 구성
void vector.FromString(const string source, const string sep=" ")	fromstring	문자열의 텍스트 데이터에서 초기화된 벡터 생성
void vector.Arange(const scalar start, const scalar stop, const scalar step=1)	arange	주어진 간격 내에서 균일한 간격의 값 생성
void matrix.Diag(const vector v, const int ndiag=0)	diag	대각 행렬 추출 또는 대각 행렬 구성
void matrix.Tri(const int rows, const int cols, const int ndiag=0)	tri	주어진 대각선 아래에 1이 있고 다른 곳에 0이 있는 행렬을 구성합니다.
void matrix.Tril(const int rows, const int cols, const scalar array[], const int ndiag=0)	tril	k번째 대각선 위의 요소가 0인 행렬의 복사본을 반환합니다.
void matrix.Triu(const int rows, const int cols, const scalar array[], const int ndiag=0)	triu	k번째 대각선 아래의 요소가 0인 행렬의 복사본을 반환합니다.
void matrix.Vander(const vector v, const int cols=-1, const bool increasing=false)	vander	방데르몽드 행렬 생성

© 2000-2025, MetaQuotes Ltd.

메서드 matrix/vector	NumPy의 Analogous 메서드	설명
vector matrix.Row(const unsigned nrow)		행 벡터 반환
vector matrix.Col(const unsigned ncol)		열 벡터 반환
unsigned matrix.Rows()		행렬의 행 수 반환
unsigned matrix.Cols()		행렬의 열 수를 반환
void matrix.Init()		행렬 초기화
matrix matrix.Transpose()	transpose	행렬의 축을 반전시키거나 치환합니다; 수정된 행렬을 반환
matrix matrix.Dot(const matrix b)	dot	두 행렬의 내적
matrix matrix.Inner(const matrix b)	inner	두 행렬의 내적
matrix matrix.Outer(const matrix b)	outer	두 행렬의 외적 계산
matrix matrix.MatMul(const matrix b)	matmul	두 행렬의 행렬 곱
matrix matrix.MatrixPower(const int power)	matrix_power	정방 행렬을 (정수) 거듭제곱 n으로 올립니다.
matrix matrix.Kron(const matrix b)	kron	두 행렬의 크로네커 곱 반환
bool matrix.Cholesky(matrix& L)	cholesky	촐레스키 분해 반환
bool matrix.QR(matrix& Q, matrix& R)	qr	행렬의 qr 분해 계산
bool matrix.SVD(matrix& U, matrix& V, vector& singular_values)	svd	특이값 분해
bool matrix.Eig(matrix& eigen_vectors, vector& eigen_values)	eig	정방 행렬의 고유값 및 오른쪽 고유 벡터 계산
bool matrix.EigH(matrix& eigen_vectors, vector& eigen_values)	eigh	에르미트 행렬의 고유값과 고유 벡터를 반환합니다.
bool matrix.EigVals(vector& eigen_values)	eigvals	일반 행렬의 고유값 계산
bool matrix.EigValsH(vector& eigen_values)	eigvalsh	에르미트 행렬의 고유값 계산
bool matrix.LU(matrix& L, matrix& U)		하부 삼각 행렬과 상부 삼각 행렬의 곱으로 행렬의 LU 분해

© 2000-2025, MetaQuotes Ltd.

메서드 matrix/vector	NumPy의 Analogous 메서드	설명
bool matrix.LUP(matrix& L, matrix& U, matrix& P)		행 순열만 있는 LU 분해를 참조하는 부분 피벗을 사용한 LUP 분해: PA=LU
double matrix.Norm(const norm)	norm	행렬 또는 벡터 norm 반환
double matrix.Cond(const norm)	cond	행렬의 조건 번호 계산
vector matrix.Spectrum()		행렬의 스펙트럼을 곱 AT*A의 고유값 집합으로 계산합니다.
double matrix.Det()	det	배열의 행렬식 계산
int matrix.Rank()	matrix_rank	가우스 방법을 사용하여 배열의 행렬 순위 반환
int matrix.SLogDet(int& sign)	slogdet	배열 행렬식의 부호와 로그 계산
double matrix.Trace()	trace	행렬의 대각선을 따라 합을 반환합니다.
vector matrix.Solve(const vector b)	solve	선형 행렬 방정식 또는 선형 대수 방정식 시스템 풀기
vector matrix.LstSq(const vector b)	lstsq	선형 대수 방정식의 최소 제곱 해를 반환합니다(비제곱 또는 축퇴 행렬의 경우).
matrix matrix.Inv()	inv	행렬의 (승법) 역행렬 계산
matrix matrix.PInv()	pinv	무어-펜로즈 방법으로 행렬의 의사 역행렬 계산
int matrix.Compare(const matrix matrix_c, const double epsilon) int matrix.Compare(const matrix matrix_c, const int digits) int vector.Compare(const vector vector_c, const double epsilon) int vector.Compare(const vector vector_c, const int digits)		지정한 정밀도로 두 행렬/벡터의 요소를 비교
double matrix.Flat(const ulong index) bool matrix.Flat(const ulong index,const double value)	flat	두 개 대신 하나의 인덱스를 통해 행렬 요소의 주소 지정 허용
double vector.ArgMax() double matrix.ArgMax() vector matrix.ArgMax(const int axis)	argmax	최대값의 인덱스를 반환
double vector.ArgMin() double matrix.ArgMin()	argmin	최소값의 인덱스를 반환

© 2000-2025, MetaQuotes Ltd.

메서드 matrix/vector	NumPy의 Analogous 메서드	설명
vector matrix.ArgMin(const int axis)		
double vector.Max() double matrix.Max() vector matrix.Max(const int axis)	max	행렬/벡터의 최대값을 반환
double vector.Mean() double matrix.Mean() vector matrix.Mean(const int axis)	mean	요소 값의 산술 평균 계산
double vector.Min() double matrix.Min() vector matrix.Min(const int axis)	min	행렬/벡터의 최소값을 반환
double vector.Sum() double matrix.Sum() vector matrix.Sum(const int axis)	sum	행렬/벡터 요소의 합을 반환. 주어진 축(축)에 대해서도 수행할 수 있음
void vector.Clip(const double min_value,const double max_value) void matrix.Clip(const double min_value,const double max_value)	clip	행렬/벡터의 요소를 유효한 값의 지정한 범위로 제한
vector vector.CumProd() vector matrix.CumProd() matrix matrix.CumProd(const int axis)	cumprod	주어진 축을 따르는 요소를 포함하여 행렬/벡터 요소의 누적 곱을 반환합니다.
vector vector.CumSum() vector matrix.CumSum() matrix matrix.CumSum(const int axis)	cumsum	주어진 축을 따르는 요소를 포함하여 행렬/벡터 요소의 누적 합계를 반환합니다.
double vector.Prod(const double initial=1) double matrix.Prod(const double initial=1) vector matrix.Prod(const int axis,const double initial=1)	prod	주어진 축에 대해서도 수행할 수 있는 행렬/벡터 요소의 곱을 반환
void matrix.Reshape(const ulong rows, const ulong cols)	reshape	데이터를 변경하지 않고 행렬의 모양 변경
void matrix.Resize(const ulong rows,const ulong cols)	resize	모양과 크기가 변경된 새 행렬을 반환
bool matrix.SwapRows(const ulong row1, const ulong row2)		행렬의 행 바꾸기
bool matrix.SwapCols(const ulong col1, const ulong col2)		행렬의 열 바꾸기

© 2000-2025, MetaQuotes Ltd.

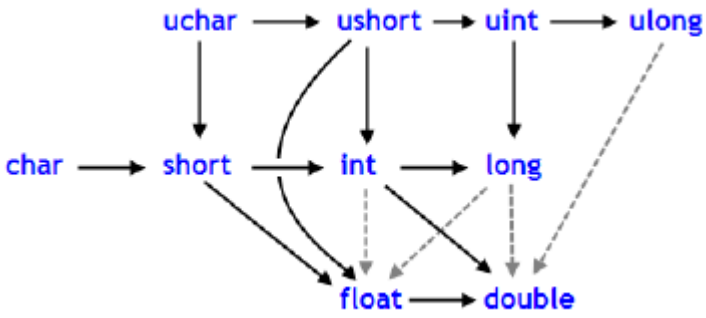
메서드 matrix/vector	NumPy의 Analogous 메서드	설명
double vector.Ptp() double matrix.Ptp() vector matrix.Ptp(const int axis)	ptp	Max() – Min()에 해당하는 행렬/벡터 또는 지정된 행렬 축의 값의 범위를 반환
double vector.Percentile(const int percent) double matrix.Percentile(const int percent) vector matrix.Percentile(const int percent,const int axis)	percentile	지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 지정된 백분위수를 반환 '퍼센트' 매개변수의 유효한 값은 [0, 100] 범위에 있습니다.
double vector.Quantile(const int percent) double matrix.Quantile(const int percent) vector matrix.Quantile(const int percent,const int axis)	quantile	지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 지정된 분위수를 반환 '퍼센트' 매개변수는 [0, 1] 범위의 값을 사용합니다.
double vector.Median() double matrix.Median() vector matrix.Median(const int axis)	median	행렬/벡터 요소의 중앙값을 계산. 중앙값은 배열/벡터 요소의 상위 절반을 요소의 하위 절반과 구분하는 중간 값입니다.
double vector.Average() double matrix.Average() vector matrix.Average(const int axis)	average	행렬/벡터 값의 산술 평균을 계산. 분모의 가중치의 합은 0이 될 수 없지만 일부 가중치는 0이 될 수 있습니다.
double vector.Std() double matrix.Std() vector matrix.Std(const int axis)	std	주어진 축을 따라 행렬/벡터 요소 또는 요소 값의 표준 편차를 반환.
double vector.Var() double matrix.Var() vector matrix.Var(const int axis)	var	행렬/벡터 요소 값의 분산을 계산
double vector.CorrCoef(const vector& v) matrix matrix.CorrCoef()	corrcoef	피어슨 상관 계수(선형 상관 계수)를 계산. 상관 계수는 [-1, 1] 범위에 있습니다.
vector vector.Correlate(const vector& v,enum mode)	correlate	두 벡터의 상호 상관을 계산. 'mode' 매개변수는 선형 컨볼루션 계산 모드를 결정합니다.
vector vector.Convolve(const vector& v, enum mode)	convolve	두 벡터의 이산 선형 컨볼루션을 반환합니다. 'mode' 매개변수는 선형 컨볼루션 계산 모드를 결정합니다.
matrix matrix.Cov() matrix vector.Cov(const vector& v); (resulting matrix 2 x 2)	cov	공분산 행렬을 계산. 두 표본(두 랜덤 변수)의 공분산은 선형 종속성을 측정한 것입니다.

© 2000-2025, MetaQuotes Ltd.

타입캐스팅

숫자 유형 캐스팅

한 숫자 유형을 다른 숫자 유형으로 변환해야 하는 경우가 종종 있습니다. 일부 숫자 유형은 다른 숫자 유형으로 변환할 수 없습니다. 다음은 허용된 캐스팅 방식입니다:



화살표가 있는 실선은 정보의 손실 없이 대부분 수행되는 변화를 나타냅니다. char 유형 대신 bool 유형을 사용(둘 다 메모리 1바이트 할애)할 수 있고, int 유형 대신 color 유형을 사용(4 바이트)할 수 있으며, long 유형 대신 datetime 유형을 사용(8 바이트)할 수 있습니다. 화살표로 표시된 4개의 파선 회색 선은 정밀도가 손실될 수 있는 변환을 나타냅니다. 예를 들어, 123456789 (int)와 같은 정수의 자릿수가 float로 나타낼 수 있는 자릿수보다 큽니다.

```
   int n=123456789;
   float f=n;     // f의 내용은 1.234567892E8 입니다
   Print("n = ",n,"   f = ",f);
   // result n= 123456789    f= 123456792.00000
```

float로 변환된 숫자는 순서는 같지만 정확도가 떨어집니다. 검은색 화살표와 달리 변환은 데이터 손실 가능성으로 이어질 수 있습니다. char와 uchar, short와 ushort, int와 uint, long과 ulong(양쪽으로의 변환) 사이의 변환은 데이터 손실을 초래할 수 있습니다.

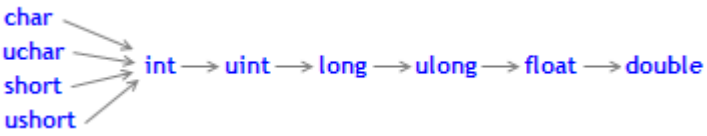
부동 소수점 값을 정수 형식으로 변환하면 분수 부분이 항상 삭제됩니다. float을 가장 가까운 정수로 반올림하려면 MathRound()를 사용해야 합니다.

예제:

```
//--- 중력 가속도
   double g=9.8;
   double round_g=(int)g;
   double math_round_g=MathRound(g);
   Print("round_g = ",round_g);
   Print("math_round_g = ",math_round_g);
/*
  결과:
   round_g = 9
   math_round_g = 10
*/
```

이진 연산자에 의해 두 값이 결합되는 경우, 연산 실행 전에 하위 유형의 피연산자가 아래 체계에서 주어진 우선순위에 따라 상위 유형으로 변환됩니다.

© 2000-2025, MetaQuotes Ltd.



데이터 유형 char, uchar, short 및 ushort는 무조건 int 유형으로 변환됩니다.

예제:

```
   char    c1=3;
//--- 첫 번째 예제
   double d2=c1/2+0.3;
   Print("c1/2 + 0.3 = ",d2);
// 결과:    c1/2+0.3 = 1.3

//--- 두 번째 예제
   d2=c1/2.0+0.3;
   Print("c1/2.0 + 0.3 = ",d2);
// 결과:    c1/2.0+0.3 = 1.8
```

계산된 식은 두 개의 연산으로 구성됩니다. 첫 번째 예제에서 문자 유형의 변수 c1은 분할 연산의 두 번째 피연산자인 상수 2가 int 유형이 높기 때문에 int 유형의 임시 변수로 변환됩니다. 정수 나눗셈 3/2의 결과로 int 타입의 값 1이 나옵니다.

첫 번째 예제의 두 번째 연산에서 두 번째 연산자는 상수 0.3이며, 이는 이중 유형이므로 첫 번째 연산 결과는 값이 1.0인 이중 유형의 임시 변수로 변환됩니다.

두 번째 예제에서 문자 유형 c1의 변수는 분할 연산의 두 번째 피연산자인 상수 2.0이 double 유형이기 때문에 더 이상 변환되지 않기 때문에 이중 유형의 임시 변수로 변환됩니다.

숫자 유형의 타입캐스팅

MQL5 언어의 표현식에서는 명시적 형식과 암시적 형식 타입캐스팅을 모두 사용할 수 있습니다. 명시적 형식 캐스팅은 다음과 같이 작성됩니다:

```
var_1 = (type)var_2;
```

표현식 또는 함수 실행 결과를 var_2 변수로 사용할 수 있습니다. 명시적 타입캐스팅의 함수 스타일 표기법도 가능합니다:

```
var_1 = type(var_2);
```

첫 번째 예에 근거하여 명시적인 타입캐스팅을 생각해 봅시다.

```
//--- 세 번째 예
   double d2=(double)c1/2+0.3;
   Print("(double)c1/2 + 0.3 = ",d2);
// 결과:    (doubl e)c1/2+0.3 = 1.80000000
```

분할 연산을 수행하기 전에 c1 변수가 double 유형에 명시적으로 캐스팅됩니다. 이제 첫 번째 피연산자를 변환한 결과 double 유형이 선택되었기 때문에 정수 상수 2가 double 유형의 값 2.0으로 캐스팅됩니다. 사실 명시적 타입캐스팅은 단항 연산입니다.

© 2000-2025, MetaQuotes Ltd.

게다가, 타입캐스팅을 하려고 할 때 결과가 허용 범위를 초과할 수 있습니다. 이 경우 절단이 발생합니다. 예:

```
   char c;
   uchar u;
   c=400;
   u=400;
   Print("c = ",c); // 결과 c=-112
   Print("u = ",u); // 결과 u=144
```

할당 작업 제외 작업을 수행하기 전에 데이터가 최대 우선 순위 유형으로 변환됩니다. 할당 작업을 수행하기 전에 데이터가 대상 유형으로 캐스팅됩니다.

예제:

```
   int    i=1/2;         // 타입캐스팅 없음, 결과는 0
   Print("i = 1/2  ",i);

   int k=1/2.0;          // 표현식은 double 형태로 캐스팅됩니다,
   Print("k = 1/2  ",k);  // 그렇다면 int의 대상 유형에 해당하며, 결과는 0입니다

   double d=1.0/2.0;     // 타입캐스팅 없음, 결과는 0.5
   Print("d = 1/2.0; ",d);

   double e=1/2.0;       // 표현식은 double 유형으로 캐스팅됩니다,
   Print("e = 1/2.0; ",e);// 대상 유형과 동일하며 결과는 0.5입니다

   double x=1/2;         // int 형식의 식이 double 타겟 typr으로 캐스팅됩니다,
   Print("x = 1/2; ",x);  // 결과는 0.0 입니다
```

long/ulong 유형을 double로 변환할 때 정수 값이 9223372036854774784보다 크거나 -9223372036854774784보다 작은 경우 정밀도가 손실될 수 있습니다.

```
void OnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- double로 캐스팅될 때 정확도가 손실되지 않는 최대 정수 값 정의
   while(l_max!=long((double)l_max))
      l_max--;
//--- double로 캐스팅될 때 정확도가 손실되지 않는 최저 정수 값 정의
   while(l_min!=long((double)l_min))
      l_min++;
//--- 정수 값에 대해 찾은 간격 도출
   PrintFormat("정수 값을 double 로 캐스팅할 때는, 그것은 반드시 "
               "[%64d, %64d] 간격 이내여야 합니다",l_min,l_max);
//--- 이제 값이 이 간격을 벗어날 경우 어떻게 되는지 살펴보겠습니다
   PrintFormat("l_max+1=%I64d, double(l_max+1)=%.f, ulong(double(l_max+1))=%I64d",
               l_max+1,double(l_max+1),long(double(l_max+1)));
   PrintFormat("l_min-1=%I64d, double(l_min-1)=%.f, ulong(double(l_min-1))=%I64d",
               l_min-1,double(l_min-1),long(double(l_min-1)));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 다음과 같은 결과를 얻습니다
// 정수 값을 double로 캐스팅할 경우 [-9223372036854774784, 9223372036854774784] 구간 내에 있
// l_max+1=9223372036854774785, double(l_max+1)=9223372036854774800, ulong(double(l_ma
// l_min-1=-9223372036854774785, double(l_min-1)=-9223372036854774800, ulong(double(l_
  }
```

문자열 유형에 대한 타입캐스팅

문자열 유형은 단순 유형 중에서 우선 순위가 가장 높습니다. 따라서 작업의 피연산자 중 하나가 문자열 유형인 경우 두 번째 피연산자는 자동으로 문자열로 캐스팅됩니다. 문자열의 경우 단일 2단계 추가 작업이 가능합니다. 숫자 유형에 문자열을 명시적으로 캐스팅할 수 있습니다.

예제:

```
   string s1=1.0/8;            // 표현식은 double 유형으로 캐스팅됩니다,
   Print("s1 = 1.0/8; ",s1);     //  그러면 문자열 타입을 타겟으로 됩니다,
// 결과는 "0.12500000" (10 개 문자를 포함하는 문자열)입니다

   string s2=NULL;             // 문자열 초기화 취소
   Print("s2 = NULL; ",s2);      // 결과는 빈 문자열입니다
   string s3="Ticket N"+12345; // 표현식이 문자열 유형에 캐스팅됩니다
   Print("s3 = \"Ticket N\"+12345",s3);

   string str1="true";
   string str2="0,255,0";
   string str3="2009.06.01";
   string str4="1.2345e2";
   Print(bool(str1));
   Print(color(str2));
   Print(datetime(str3));
   Print(double(str4));
```

파생 클래스의 포인터에 대한 기본 클래스 포인터의 타입캐스팅

open generated 클래스의 개체는 해당 기본 클래스의 개체로도 볼 수 있습니다. 이것은 몇 가지 흥미로운 결과로 이어집니다. 예를 들어 단일 기본 클래스에서 생성된 서로 다른 클래스의 개체가 서로 크게 다를 수 있지만 기본 유형의 개체로 볼 때 연결된 목록(List)을 만들 수 있습니다. 그러나 기본 클래스 개체는 자동으로 파생 클래스의 개체가 아닙니다.

명시적 캐스팅을 사용하여 기본 클래스 포인터를 파생 클래스의 포인터로 변환할 수 있습니다. 그러나 이러한 변환의 허용 가능성에 대해 완전히 확신해야 합니다. 그렇지 않으면 심각한 런타임 오류가 발생하고 mql5 프로그램이 중지되기 때문입니다.

dynamic_cast 연산자를 사용한 동적 타입캐스팅

© 2000-2025, MetaQuotes Ltd.

동적 타입캐스팅은 클래스에 대한 포인터에만 적용할 수 있는 dynamic_cast 연산자를 사용하여 수행됩니다. 유형 유효성 검사가 런타임에 수행됩니다. 즉, 컴파일러는 dynamic_cast 연산자를 사용할 때 형식 캐스팅에 적용된 데이터 형식을 확인하지 않습니다. 포인터가 실제 개체 유형이 아닌 데이터 유형으로 변환될 경우 결과는 NULL이 됩니다.

```
dynamic_cast <type-id> ( expression )
```

꺽쇠괄호의 *type-id* 매개변수는 이전에 정의된 클래스 유형을 가리켜야 합니다. C++와는 달리, *표현식*n 피연산자 유형은 void를 제외한 모든 값을 가질 수 있습니다.

예제:

```
class CBar { };
class CFoo : public CBar { };

void OnStart()
  {
   CBar bar;
//--- *foo 포인터에 *bar 포인터 유형을 동적 캐스팅할 수 있습니다.
   CFoo *foo = dynamic_cast<CFoo *>(&bar); // 치명적 오류 없음
   Print(foo);                             // foo=NULL
//--- Foo type 객체에 대한 Bar type 객체 참조를 명시적으로 캐스팅하려는 시도는 금지됩니다
   foo=(CFoo *)&bar;                       // 치명적 런타임 오류
   Print(foo);                             // 이 문자열은 실행되지 않습니다
  }
```

더 보기

데이터 타입

© 2000-2025, MetaQuotes Ltd.

Void 타입 및 NULL 상수

구문적으로 void 유형은 char, uchar, bool, short, ushort, int, uint, color, long, ulong, datetime, float, double 및 string 유형과 함께 기본적인 유형입니다. 이 유형은 함수가 값을 반환하지 않음을 나타내기 위해 사용하거나 함수 매개 변수로 매개 변수가 없음을 나타냅니다.

미리 정의된 상수 변수인 **NULL**이 *void* 타입입니다. 변환 없이 다른 기본 유형의 변수에 할당할 수 있습니다. 기본 유형의 변수를 **NULL** 값과 비교할 수 있습니다.

예제:

```
//--- 문자열이 초기화되지 않은 경우 사전 정의된 값을 지정합니다
if(some_string==NULL) some_string="empty";
```

또한 **NULL**을 new 연산자로 만든 개체에 대한 포인터와 비교할 수 있습니다.

더 보기

변수, 함수

© 2000-2025, MetaQuotes Ltd.

사용자정의 유형

typedef 키워드(C++)를 사용하면 사용자 정의 데이터 유형을 만들 수 있습니다. 이렇게 하려면 기존 데이터 유형에 대해 새 데이터 유형 이름을 지정하기만 하면 됩니다. 새 데이터 유형이 생성되지 않았습니다. 대신 기존 유형의 새 이름이 정의됩니다. 사용자 정의 유형을 사용하면 응용 프로그램의 유연성이 향상됩니다. 때로는 대체 매크로(#define)를 사용하여 유형 정의 지침을 변경해도 충분합니다. 사용자 정의 유형을 사용하면 typedef를 사용하여 표준 데이터 유형에 사용자 정의 이름을 적용할 수 있으므로 코드 가독성도 향상됩니다. 사용자 정의 유형을 작성하기 위한 항목의 일반 형식:

```
   typedef type new_name;
```

여기서, *유형*은 허용되는 모든 데이터 유형을 의미하며 *new_name*은 유형의 새 이름입니다. 새 이름은 기존 유형 이름에 대한 추가(대체가 아님)로만 설정됩니다. MQL5를 사용하면 typedef를 사용하여 함수에 대한 포인터를 만들 수 있습니다.

함수에 대한 포인터

함수에 대한 포인터는 일반적으로 다음 형식으로 정의됩니다

```
   typedef function_result_type (*Function_name_type)(list_of_input_parameters_types);
```

여기서 typedef 다음에 함수 시그니처(입력 파라미터의 수와 유형, 함수에 의해 반환되는 결과 유형)이 설정됩니다. 다음은 포인터를 만들고 함수에 적용하는 간단한 예입니다:

```
//--- 두 개의 int 매개변수를 수신하는 함수에 대한 포인터를 선언합니다
   typedef int (*TFunc)(int,int);
//--- TFunc는 유형이며 함수에 대한 변수 포인터를 선언할 수 있습니다
   TFunc func_ptr; // 함수에 대한 포인터
//--- TFunc 설명에 해당하는 함수를 선언합니다
   int sub(int x,int y) { return(x-y); }  // 수를 빼기
   int add(int x,int y) { return(x+y); }  // 두 수의 합
   int neg(int x)       { return(~x);  }  // 변수의 비트 반전
//--- func_ptr 변수는 함수 주소를 저장하여 나중에 선언할 수 있습니다
   func_ptr=sub;
   Print(func_ptr(10,5));
   func_ptr=add;
   Print(func_ptr(10,5));
   func_ptr=neg;           // 오류: neg에 int(int,int) 유형이 없습니다
   Print(func_ptr(10));    // 오류: 두 개의 매개 변수가 필요합니다
```

이 예제에서, *func_ptr* 변수는 *sub* 및 *add* 함수를 수신할 수 있는데, 이는 int 유형의 두 인수가 함수에 대한 *TFunc* 포인터에 있기 때문입니다. 반대로 *neg* 함수는 시그니처가 다르기 때문에 func_ptr 포인터에 할당할 수 없습니다.

사용자 인터페이스에서 이벤트 모델 정렬

함수에 대한 포인터를 사용하면 사용자 인터페이스를 만들 때 이벤트 처리를 쉽게 만들 수 있습니다. CButton 섹션의 예를 사용하여 버튼을 만들고 버튼을 누르는 기능을 추가해 보겠습니다. 먼저 버튼을 눌러 호출할 *TAction* 함수에 대한 포인터를 정의하고 다음에 따라 세 가지 함수를 만듭니다: *TAction 설명.*

```
//--- 사용자 정의 함수 유형 만들기
typedef int(*TAction)(string,int);
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|  파일 열기                                                       |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("%s 함수 호출됨 (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  파일 저장                                                       |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("%s 함수 호출됨 (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  파일 닫기                                                       |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("%s 함수 호출됨 (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
```

그런 다음 CButton에서 MyButton 클래스를 만들고, 여기서 *TAction* 포인터를 함수에 추가해야 합니다.

```
//+------------------------------------------------------------------+
//| 이벤트 처리 기능으로 버튼 클래스 만들기                          |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // 차트 이벤트 핸들러
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- 이벤트 처리 함수에 대한 버튼 텍스트 및 포인터 지정하는 생성자
                     MyButton(string text, TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- OnEvent() 이벤트 처리기에서 호출된 사용자 지정 함수 설정
   void              SetAction(TAction act){m_action=act;}
   //--- 표준 차트 이벤트 처리기
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      if(m_action!=NULL && lparam==Id())
        {
         //--- 사용자 지정 m_action() 핸들러를 호출합니다.
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- CButton 부모 클래스에서 처리기를 호출한 결과를 반환합니다
         return(CButton::OnEvent(id,lparam,dparam,sparam));
    }
  };
```

CAppDialog에서 CControlsDialog 파생 클래스를 만들고 *m_buttons* 배열을 추가하는데 이는 *MyButton* 유형의 버튼과 다음을 저장하기 위함입니다: *AddButton(MyButton &button)* 및 *CreateButtons() methods*.

```
//+------------------------------------------------------------------+
//| CControlsDialog 클래스                                           |
//| 목적: 어플리케이션 관리를 위한 그래픽 패널                       |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // 버튼 배열
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 버튼 추가
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- 버튼 생성
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| 차트에 CControlsDialog 객체 생성                                 |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
  }
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 들여쓰기 및 간격
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계선 너비 허용
```

© 2000-2025, MetaQuotes Ltd.

```
#define INDENT_TOP                          (11)      // 위에서 들여쓰기(경계선 너비 허용)
#define CONTROLS_GAP_X                      (5)       // 간격(X 좌표)
#define CONTROLS_GAP_Y                      (5)       // 간격(Y 좌표)
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // 크기(X 좌표)
#define BUTTON_HEIGHT                       (20)      // 크기(Y 좌표)
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // 크기(Y 좌표)
//+------------------------------------------------------------------+
//| CControlsDialog 패널에 버튼 작성 및 추가                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- 버튼 좌표 계산
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- 버튼 개체와 함수에 대한 포인터 추가
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",::Save));
   AddButton(new MyButton("Close",Close));
//--- 버튼을 그래픽으로 생성
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("버튼 생성 실패 %s %d",b.Text(),i);
         return(false);
        }
      //--- 각 버튼을 CControlsDialog 컨테이너에 추가
      if(!Add(b))
         return(false);
     }
//--- 성공
   return(true);
  }
```

이제 3개의 버튼이 있는 CControlsDialog 제어판을 사용하여 프로그램을 개발할 수 있습니다: 열기, 저장 및 닫기. 버튼을 클릭하면 *TAction* 포인터 형태의 해당 함수가 호출됩니다.

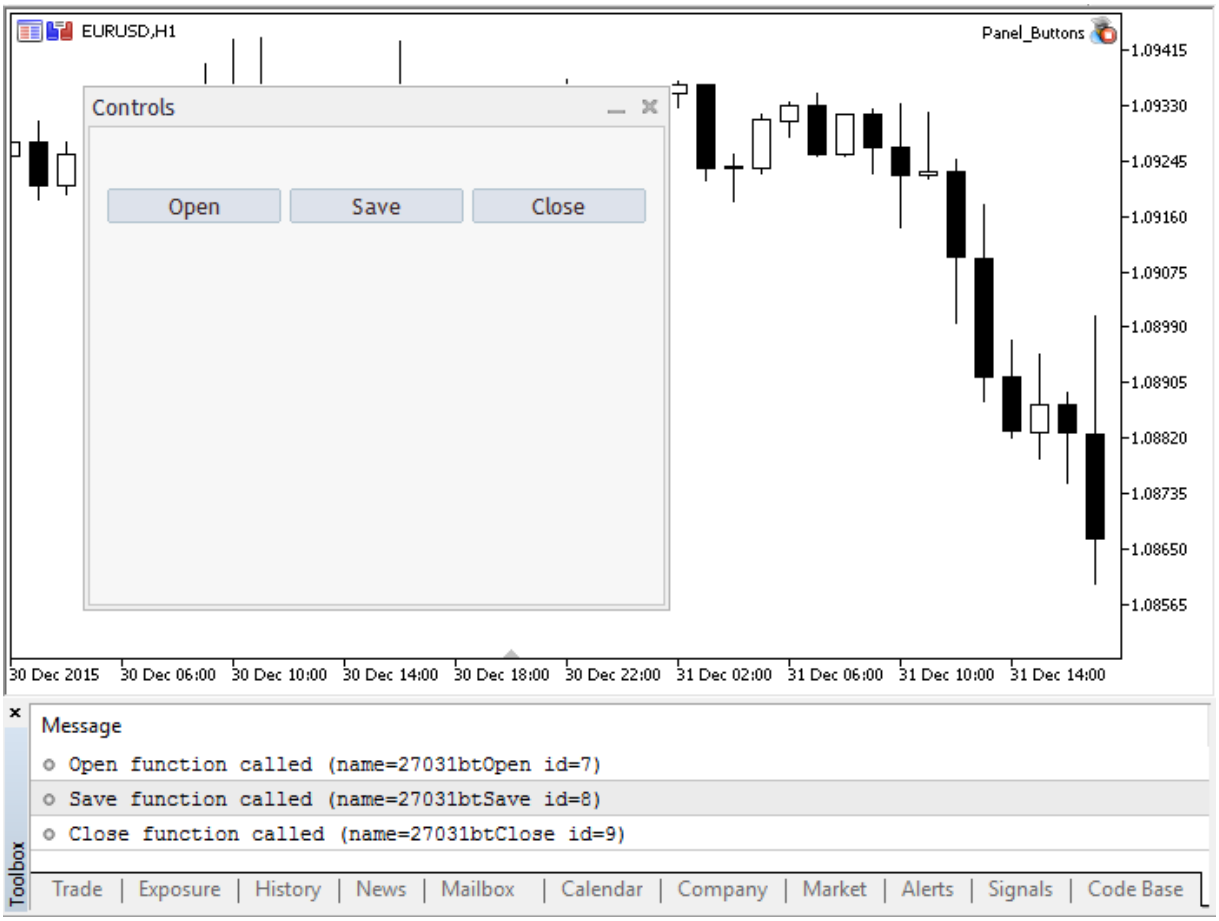
```
//--- 프로그램을 시작할 때 개체를 자동으로 만들려면 전역 수준에서 개체를 선언합니다
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- 이제 차트에 개체를 만듭니다
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   MyDialog.Run();
//--- 어플리케이션이 성공적으로 초기화됨
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 대화상자 제거
   MyDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // 문자열 타입의 이벤트 매개 변수
  {
//--- 차트 이벤트에 대해 부모 클래스에서 핸들러를 호출(여기서는 CAppDialog 입니다).
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

실행된 응용 프로그램의 모양과 버튼 클릭 결과가 스크린샷에 제공됩니다.

© 2000-2025, MetaQuotes Ltd.



프로그램의 전체 소스 코드

```
//+------------------------------------------------------------------+
//|                                                Panel_Buttons.mq5 |
//|                        Copyright 2017, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+

#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "The panel with several CButton buttons"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 들여쓰기 및 간격
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계선 너비 허용
#define INDENT_TOP                          (11)      // 위에서 들여쓰기(경계선 너비 허용)
#define CONTROLS_GAP_X                      (5)       // 간격(X 좌표)
#define CONTROLS_GAP_Y                      (5)       // 간격(Y 좌표)
```

```
//--- 버튼용
#define BUTTON_WIDTH                    (100)      // 크기(X 좌표)
#define BUTTON_HEIGHT                   (20)       // 크기(Y 좌표)
//--- 표시 영역용
#define EDIT_HEIGHT                     (20)       // 크기(Y 좌표)

//--- 사용자 정의 함수 유형 생성
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  파일 열기                                                       |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("%s 함수 호출됨 (name=%s id=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  파일 저장                                                       |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("%s 함수 호출됨 (name=%s id=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  파일 닫기                                                       |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("%s 함수 호출됨 (name=%s id=%d)",__FUNCTION__,name,id);
   return(3);
  }
//+------------------------------------------------------------------+
//| 이벤트 처리 기능으로 버튼 클래스 만들기       |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // 차트 이벤트 핸들러
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- 이벤트 처리 함수에 대한 버튼 텍스트 및 포인터 지정하는 생성자
                     MyButton(string text,TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- OnEvent() 이벤트 처리기에서 호출된 사용자 지정 함수 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   void              SetAction(TAction act){m_action=act;}
   //--- 표준 차트 이벤트 처리기
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- 커스텀 핸들러 호출
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- CButton 부모 클래스에서 처리기를 호출한 결과를 반환합니다
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
//+------------------------------------------------------------------+
//| CControlsDialog 클래스                                           |
//| 목적: 어플리케이션 관리를 위한 그래픽 패널                       |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // 버튼 배열
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 버튼 추가
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- 버튼 생성
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| 차트에 CControlsDialog 객체 생성                                 |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
   return(CreateButtons());
//---
  }
//+------------------------------------------------------------------+
//| CControlsDialog 패널에 버튼 작성 및 추가                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 버튼 좌표 계산
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- 버튼 개체와 함수에 대한 포인터 추가
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",::Save));
   AddButton(new MyButton("Close",Close));
//--- 버튼을 그래픽으로 생성
   for(int i=0;i<m_buttons.Total();i++)
     {
      MyButton *b=(MyButton*)m_buttons.At(i);
      x1=INDENT_LEFT+i*(BUTTON_WIDTH+CONTROLS_GAP_X);
      x2=x1+BUTTON_WIDTH;
      if(!b.Create(m_chart_id,m_name+"bt"+b.Text(),m_subwin,x1,y1,x2,y2))
        {
         PrintFormat("버튼 생성 실패 %s %d",b.Text(),i);
         return(false);
        }
      //--- 각 버튼을 CControlsDialog 컨테이너에 추가
      if(!Add(b))
         return(false);
     }
//--- 성공
   return(true);
  }
//--- 프로그램을 시작할 때 개체를 자동으로 만들려면 전역 수준에서 개체를 선언합니다
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 이제 차트에 개체를 만듭니다
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   MyDialog.Run();
//--- 어플리케이션이 성공적으로 초기화됨
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 대화상자 제거
   MyDialog.Destroy(reason);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // 이벤트 ID
                  const long& lparam,    // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // 문자열 타입의 이벤트 매개 변수
  {
//--- 차트 이벤트에 대해 부모 클래스에서 핸들러를 호출(여기서는 CAppDialog 입니다).
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

더 보기

변수, 함수

© 2000-2025, MetaQuotes Ltd.

객체 포인터

MQL5에서는 복잡한 유형 객체의 동적 생성이 가능합니다. 이것은 생성된 객체의 설명자를 반환하는 'new' 연산자를 사용하여 수행됩니다. 설명자의 사이즈는 8바이트입니다. MQL5의 객체 설명자는 구문적으로 C++ 포인터와 유사합니다.

예:

```
MyObject* hobject= new MyObject();
```

C++와 달리 위 예제의 'hobject' 변수는 메모리에 대한 포인터가 아니라 객체 설명자입니다. 또한 MQL5에서 함수에서 매개변수의 모든 객체는 참조로 전달되어야 합니다. 아래 예는 객체를 함수 매개변수로 전달하는 것을 보여줍니다:

```
class Foo
  {
public:
   string            m_name;
   int               m_id;
   static int        s_counter;
 //--- 생성자와 소멸자
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
   //--- Foo 객체 초기화
   void              Setup(string name)
     {
      m_name=name;
      s_counter++;
      m_id=s_counter;
     }
  };
int Foo::s_counter=0;
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
void OnStart()
  {
//--- 자동 생성으로 객체를 변수로 선언
   Foo foo1;
//--- Transform passing object by reference
   PrintObject(foo1);

//--- 객체에 대한 포인터를 선언하고 'new' 연산자를 사용하여 생성
   Foo *foo2=new Foo("foo2");
//--- 참조로 객체에 대한 포인터를 전달하는 변형
   PrintObject(foo2); // 객체에 대한 포인터는 컴파일러에 의해 자동으로 변환됩니다

//--- Foo 객체의 배열 선언
   Foo foo_objects[5];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 객체 배열 전달의 변형
   PrintObjectsArray(foo_objects); // 객체 배열을 전달하기 위한 별도의 함수

//--- Foo 유형의 객체에 대한 포인터 배열 선언
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
      foo_pointers[i]=new Foo("foo_pointer");
//--- 포인터의 배열 전달의 변형
   PrintPointersArray(foo_pointers); // 포인터 배열을 전달하기 위한 별도의 함수

//--- 완료하기 전에 포인터로 생성된 객체를 삭제해야 합니다
   delete(foo2);
//--- 포인터의 배열을 제거
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
/+------------------------------------------------------------------+
//|  객체는 항상 참조로 전달됩니다.                                  |
/+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," Object name=",object.m_name);
  }
/+------------------------------------------------------------------+
//| 객체의 배열을 전달                                               |
/+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
/+------------------------------------------------------------------+
//| 객체 포인터의 배열을 전달                                        |
/+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
/+------------------------------------------------------------------+
```

사용하기 전에 포인터를 체크하세요

© 2000-2025, MetaQuotes Ltd.

잘못된 포인터에 액세스하려는 시도는 프로그램의 치명적 종료로 이어집니다. CheckPointer 함수는 포인터를 사용하기 전에 확인하는 기능입니다. 포인터는 다음과 같은 경우에 유효하지 않을 수 있습니다:

- 포인터가 NULL; 일때
- 객체가 delete 연산자를 사용하여 제거 되었을 때

이 함수는 포인터의 유효성을 검사하는 데 사용될 수 있습니다. 0이 아닌 값은 이 포인터에서 데이터에 액세스할 수 있음을 나타냅니다.

```
class CMyObject
 {
protected:
  double             m_value;
public:
                     CMyObject(void);
                     CMyObject(double value) {m_value=value;};
                    ~CMyObject(void){};
  //---
  double             Value(void) {return(m_value);}
 };
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
void OnStart()
 {
//--- 초기화되지 않은 객체 생성
  CMyObject *pointer;
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- 포인터 초기화
  pointer=new CMyObject(M_PI);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("2. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- 객체를 삭제
  delete(pointer);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("3. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. pointer is POINTER_INVALID
   2. pointer.Value()=3.141592653589793
   3. pointer is POINTER_INVALID
```

© 2000-2025, MetaQuotes Ltd.

```
*/
```

포인터의 유효성을 빠르게 확인하려면 "!" 연산자를 사용할 수도 있습니다. (LNOT) CheckPointer 함수의 묵시적 호출을 통해 이를 확인합니다. 이는 좀더 깔끔하고 간략한 코드 작성을 가능하게 합니다. 아래는 이전 예에서 체크한 것입니다:

```
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
void OnStart()
 {
//--- 초기화되지 않은 객체 생성
  CMyObject *pointer;
  if(!pointer)
    Print("1. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=", pointer.Value());

//--- 포인터 초기화
  pointer=new CMyObject(M_PI);
  if(!pointer)
    Print("2. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=", pointer.Value());

//--- 객체를 삭제
  delete(pointer);
  if(!pointer)
    Print("3. pointer is ", EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=", pointer.Value());
 }
/*
   1. pointer is POINTER_INVALID
   2. pointer.Value()=3.141592653589793
   3. pointer is POINTER_INVALID
*/
```

연산자 "=="는 NULL을 빠르게 확인하는 데 사용됩니다. 일례: ptr==NULL 혹은 ptr!=NULL.

더 보기

변수, 변수 초기화, 가시성 범위 및 변수 수명, 객체 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

참조: 수식어 & 키워드 this

참조로 매개 변수 전달

MQL5에서 simple 유형의 매개 변수는 값과 참조를 모두 전달할 수 있지만, compound 유형의 매개 변수는 항상 참조를 통해 전달됩니다. 컴파일러에게 매개 변수가 참조를 통해 전달되어야 함을 알리려면 매개 변수 이름 앞에 ampersand 문자 &가 추가됩니다.

참조로 매개 변수를 전달한다는 것은 변수의 주소를 전달하는 것을 의미하며, 따라서 참조로 전달되는 매개 변수의 모든 변경 사항이 소스 변수에 즉시 반영됩니다. 참조를 통과하는 매개변수를 사용하여 함수의 여러 결과에 대한 반환을 동시에 구현할 수 있습니다. 참조에서 전달된 매개변수의 변경을 방지하려면 const 한정자를 사용하십시오.

따라서 함수의 입력 매개변수가 구조체 또는 클래스 객체인 array인 경우, 기호 '&'는 변수 형식과 이름 앞에 함수 헤더에 배치됩니다.

예제

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
  };
//+------------------------------------------------------------------+
//| 배열 채우기                                                      |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array, array);
     }
  }
```

위의 예에서는 class CDemoClass가 선언되었으며, 여기에는 private 멤버 – double 유형의 m_array[] 배열이 포함됩니다. 함수 setArray() 가 선언되었으며, 이 함수에는 array[] 가 레퍼런스를 통해 전달됩니다. 함수 헤더에 참조 통과에 대한 표시가 없는 경우(예: 앰퍼샌드 문자가 없는 경우), 이러한 코드를 컴파일하려고 할 때 오류 메시지가 생성됩니다.

어레이가 참조를 통해 전달되지만 한 어레이를 다른 어레이에 할당할 수는 없습니다. 원본 배열의 내용을 요소별로 수신자 배열에 복사해야 합니다. 함수 설명에 &가 있는 것은 함수 매개변수로 전달될 때 배열 및 구조물에 대한 필수 조건입니다.

키워드 this

© 2000-2025, MetaQuotes Ltd.

클래스 유형(개체)의 변수는 참조 및 포인트를 통해 전달될 수 있습니다. 레퍼런스뿐만 아니라 포인터를 사용하면 객체에 액세스할 수 있습니다. 객체 포인터 선언 후에는 new 연산자를 사용하여 포인터를 만들고 초기화해야 합니다.

유보어인 **this**는 클래스 또는 구조 메서드 내에서 사용할 수 있는 개체의 참조를 얻기 위한 것입니다. **this**는 항상 개체를 사용하는 방법으로 참조하며 GetPointer(this) 식은 GetPointer() 호출을 수행하는 함수인 개체의 포인터를 제공합니다. MQL5에서 함수는 객체를 반환할 수 없지만 객체 포인터를 반환할 수 있습니다.

따라서 개체를 반환하는 함수가 필요한 경우 GetPointer(this) 형식으로 이 개체의 포인터를 반환할 수 있습니다. 이 클래스의 개체에 대한 포인터를 반환하는 getDemoClass() 함수를 CDemoClass의 설명에 추가하겠습니다.

```
class CDemoClass
  {
private:
   double            m_array[];

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------+
//| 배열 채우기                                                      |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------+
//| 자체 포인터를 반환합니다                                         |
//+------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

구조에 포인터가 없고, 연산자 *new* 및 *삭제*를 적용할 수없으며, GetPointer(this)를 사용할 수 없습니다.

더 보기

객체 포인터, 객체 생성 및 삭제, 가시성 범위 및 변수 수명

© 2000-2025, MetaQuotes Ltd.

연산 및 수식

일부 캐릭터와 캐릭터 시퀀스는 특히 중요합니다. 다음은 작업 기호라고 하는 예입니다.

```
+ - * / %      산술 연산의 기호
&& ||          논리 연산의 기호
= += *=        문자 대입 연산자
```

연산 기호는 표현식에 사용되며 적절한 피연산자가 주어졌을 때 의미가 있습니다. 문장 부호도 강조됩니다. 괄호, 중괄호, 쉼표, 콜론 및 세미콜론입니다.

연산 기호, 문장 부호 및 공백은 언어 요소를 서로 구분하는 데 사용됩니다.

이 절에는 다음 항목에 대한 설명이 포함되어 있습니다:

- 수식
- 산술 연산
- 대입 연산
- 관계 연산
- 불(논리) 연산
- 비트 연산
- 기타 연산
- 우선순위 및 연산 순서

© 2000-2025, MetaQuotes Ltd.

수식

식은 하나 이상의 피연산자와 연산 기호로 구성됩니다. 표현식은 여러 줄로 쓸 수 있습니다.

예제:

```
a++; b = 10;          // 여러 식이 한 줄에 있습니다.
//---  한 개의 식이 여러 줄로 나뉩니다
x = (y * z) /
    (w + 2) + 127;
```

세미콜론(;)으로 끝나는 식은 연산자입니다.

더 보기

우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

산술 연산

산술 연산에는 다음과 같은 덧셈 연산과 곱셈 연산이 포함됩니다:

```
변수의 합                          i = j + 2;
변수의 차                    i = j - 3;
부호 변경                          x = - x;
변수의 곱                      z = 3 * x;
나눗셈 몫                          i = j / 5;
나눗셈 나머지                      minutes = time % 60;
변수 값에 1 더하기              i++;
변수 값에 1 더하기              ++i;
변수 값에 1 빼기    k--;
변수 값에 1 빼기    --k;
```

증분 및 감소 연산은 변수에만 적용되므로 상수에 적용할 수 없습니다. 사전 증가(++i) 및 감소(—k)는 이 변수가 식에 사용되기 직전에 변수에 적용됩니다.

사후 증가(i++) 및 사후 감소(k—)는 이 변수가 식에 사용된 직후 변수에 적용됩니다.

Important Notice

```
int i=5;
int k = i++ + ++i;
```

위의 식을 한 프로그래밍 환경에서 다른 프로그래밍 환경으로 이동하는 동안(예: Borland C++에서 MQL5로) 계산 문제가 발생할 수 있습니다. 일반적으로 계산 순서는 컴파일러 구현에 따라 달라집니다. 실무적으로 사후 감소(후 증가)를 이행하는 두 가지 방법이 있습니다.

1. 전체 식을 계산한 후 변수에 사후 감소(사후 증가)가 적용됩니다.
2. 연산 시 변수에 즉시 사후 감소(사후 증가)가 적용됩니다..

현재 MQL5에서는 사후 감소(사후 증가) 계산의 첫 번째 방법이 구현되어 있습니다. 그러나 이러한 특성을 알고 있더라도 사용을 실험하는 것은 권장되지 않습니다.

예제:

```
int a=3;
a++;             // 유효한 수식
int b=(a++)*3;   // 잘못된 수식
```

더 보기

우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

대입 연산

지정된 작업을 포함하는 식의 값은 할당 후 왼쪽 피연산자 값입니다.

```
y 변수에 x 값 대입                                    y = x;
```

다음 연산은 산술 또는 비트 연산과 할당 연산을 결합합니다:

```
y 변수에 x 추가                                              y += x;
y 변수에서 x 빼기                                          y -= x;
y 변수에 x 곱하기                                            y *= x;
y 변수를 x로 나누기                                                y /= x;
y변수를 x로 나눈 나머지                                      y %= x;
y의 이진수 표현을 오른쪽으로 x비트 이동                  y >>= x;
y의 이진수 표현을 x비트로 왼쪽으로 이동                   y <<= x;
y와 x의 이진수 표현의 비트 연산                        y &= x;
y 및 x의 이진수 표현의 OR 비트 연산                          y |= x;
y 및 x의 이진수 표현의 제외 OR 비트 연산             y ^= x;
```

비트 방향 연산은 정수에만 적용할 수 있습니다. y 표현의 논리적 이동을 x비트로 오른쪽/왼쪽으로 수행할 때 x 값 중 가장 작은 5개의 이진수가 사용되고 가장 높은 자리가 삭제됩니다. 즉, 0-31비트로 전환됩니다.

%= 연산(x의 모듈별 y 값)에 의해 결과 부호는 분할된 숫자의 부호와 같습니다.

대입 연산자는 식에서 여러 번 사용할 수 있습니다. 이 경우 표현식의 처리가 왼쪽에서 오른쪽으로 수행됩니다:

```
y=x=3;
```

먼저 변수 x에 값 3이 할당되고 y 변수에는 x의 값(즉, 3)이 할당됩니다.

더 보기

우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

관계 연산

부울 FALSE는 정수 0 값으로 표시되는 반면, 부울 TRUE는 0이 아닌 값으로 표시됩니다.

관계 연산 또는 논리 연산을 포함하는 식의 값은 FALSE(0) 또는 TRUE(1)입니다.

```
a가 b와 같으면 참입니다                    a == b;
a가 b와 같지 않으면 참입니다                a != b;
a가 b보다 작으면 참입니다                   a < b;
a가 b보다 크면 참입니다                    a > b;
a가 b보다 작거나 같으면 참입니다            a <= b;
a가 b보다 크거나 같으면 참입니다           a >= b;
```

두 개의 실수의 동일성은 비교할 수 없습니다. 대부분의 경우, 15번째 소수점의 값이 다르기 때문에 동일한 것처럼 보이는 두 숫자가 동일하지 않을 수 있습니다. 두 실수를 올바르게 비교하려면 정규화된 차이를 0과 비교합니다.

예제:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double first=0.3;
   double second=3.0;
   double third=second-2.7;
   if(first!=third)
     {
      if(CompareDoubles(first,third))
         printf("%16f 과 %16f 는 같습니다",first,third);
     }
  }
// Result: 0.3000000000000000 0.2999999999999998 은 같음
```

더 보기

우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

불(논리) 연산

논리 부정 NOT (!)

논리 부정의 피연산자 (!) 는 산술 형식이어야 합니다. 피연산자 값이 FALSE(0)이면 TRUE(1)이고 피연산자가 FALSE(0)와 다르면 FALSE(0)의 결과가 나옵니다.

```
if(!a) Print("not 'a'");
```

논리 연산 OR (||)

x와 y값의 논리 OR 연산 (||). x 또는 y 값이 true(null이 아님)이면 표현식 값은 TRUE(1)입니다. 그렇지 않으면 -FALSE (0).

```
if(x<0 || x>=max_bars) Print("out of range");
```

논리 연산 AND (&&)

x와 y값의 논리 연산 AND (&&). x 및 y 값이 null이 아닌 True이면 표현식 값은 TRUE(1)입니다. 그렇지 않으면 -FALSE (0).

부울 연산의 간략한 추정치

이른바 "간단 추정치"의 체계는 부울 연산에 적용됩니다. 즉, 식의 결과를 정확하게 추정할 수 있을 때 식의 계산이 종료됩니다.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 간략한 추정의 첫 번째 예
   if(func_false() && func_true())
     {
      Print("연산 &&: 이 표현은 볼 수 없을 것입니다");
     }
   else
     {
      Print("연산 &&: 첫 번째 식 결과가 거짓이므로 두 번째 식이 계산되지 않았습니다");
     }
//--- 간략한 추정치의 두 번째 예
   if(!func_false() || !func_true())
     {
      Print("연산 ||: 첫 번째 식 결과가 참이므로 두 번째 식이 계산되지 않았습니다");
     }
   else
     {
      Print("연산 ||: 이 연산을 볼 수 없을 것입니다");
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 이 함수는 항상 false를 반환합니다                                  |
//+------------------------------------------------------------------+
bool func_false()
  {
   Print("함수 func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| 이 함수는 항상 true를 반환합니다                                   |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("함수 func_true()");
   return(true);
  }
```

더 보기

우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

비트 연산

하나로 보완

변수 값을 최대 1까지 보완합니다. 수식 값은 변수 값이 0을 포함하는 모든 자릿수에 1을 포함하며 변수가 1을 포함하는 모든 자릿수에 0을 포함합니다.

```
b = ~n;
```

예제:

```
   char a='a',b;
   b=~a;
   Print("a = ",a, "  b = ",b);
// 결과는 다음과 같습니다:
// a = 97   b = -98
```

우측 이동

x의 이진 표현은 y 자릿수로 오른쪽으로 이동합니다. 이동할 값이 부호 없는 유형인 경우 논리적 오른쪽 이동이 이루어집니다. 즉, 자유 왼쪽 비트는 0으로 채워집니다.

이동할 값이 부호 유형인 경우 산술 오른쪽 이동이 이루어집니다. 즉, 왼쪽 자유 자리가 부호 비트 값으로 채워집니다(숫자가 양수이면 부호 비트 값은 0이고, 숫자가 음수이면 부호 비트 값은 1입니다).

```
x = x >> y;
```

예제:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
//--- 우측으로 이동
   b=a>>1;
   Print("After:   a = ",a, "  b = ",b);
// 결과는 다음과 같습니다:
// Before:  a = 97   b = 98
// After:   a = 97   b = 48
```

좌측 이동

x의 이진 표현은 y 자릿수로 왼쪽으로 이동하며, 오른쪽 자유 자릿수는 0으로 채워집니다.

```
x = x << y;
```

예제:

```
   char a='a',b='b';
   Print("Before:  a = ",a, "  b = ",b);
//--- 좌측으로 이동
   b=a<<1;
   Print("After:   a = ",a, "  b = ",b);
// 결과는 다음과 같습니다:
// Before:  a = 97   b = 98
```

© 2000-2025, MetaQuotes Ltd.

```
// After:   a = 97   b = -62
```

이동된 변수의 길이와 같거나 큰 비트 수로 이동하지 않는 것이 좋습니다. 이러한 작업의 결과가 정의되지 않았기 때문입니다.

Bitwise AND 연산

이진 코드 x 및 y 표현의 비트 AND 연산입니다. 식 값은 모든 자릿수에 1(TRUE)을 포함하며, x와 y 모두 0이 아닌 숫자를 포함하며, 다른 모든 자리에는 0(FALSE)을 포함합니다.

```
b = ((x & y) != 0);
```

예제:

```
   char a='a',b='b';
//--- AND 연산
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// 결과는 다음과 같습니다:
// a = 97   b = 98
// a & b = 96
```

Bitwise OR 연산

x와 y의 이진 표현의 비트 OR 연산입니다. 식 값은 모든 자릿수에 1을 포함하며, 여기서 x 또는 y는 0을 포함하지 않고 다른 모든 자리에는 0을 포함합니다.

```
b = x | y;
```

예제:

```
   char a='a',b='b';
//--- OR 연산
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// 결과는 다음과 같습니다:
// a = 97   b = 98
// a | b = 99
```

Bitwise 전용 연산 OR

x와 y의 이진 표현의 비트 배타적 OR(eXclusive OR) 연산입니다. 식 값은 x와 y의 이진수 값이 서로 다른 모든 자릿수에 1을 포함하며, 다른 모든 자리에는 0을 포함합니다.

```
b = x ^ y;
```

예제:

```
   char a='a', b='b';
//--- 제외 OR 연산
   char c=a^b;
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("a = ",a,"  b = ",b);
   Print("a ^ b = ",c);
// 결과는 다음과 같습니다:
// a = 97   b = 98
// a ^ b = 3
```

비트 연산은 정수로만 수행됩니다.

더 보기

우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

기타 연산

인덱싱 ([])

배열의 i번째 요소를 지정할 때 표현식 값은 일련 번호 i를 가진 변수의 값입니다.

예제:

```
array[i] = 3; // 배열의 i 번째 요소에 값 3을 할당합니다.
```

정수만 배열의 인덱스가 될 수 있습니다. 4차원 이하 배열이 허용됩니다. 각 차원은 0부터 **차원 크기**-1까지 인덱싱됩니다. 특히, 50개의 요소로 구성된 1차원 배열의 경우, 첫 번째 요소에 대한 참조는 array [0]처럼 보이고 마지막 요소에 대한 참조는 array [49]입니다.

어레이를 벗어나 주소를 지정하면 실행 중인 하위 시스템이 심각한 오류를 생성하고 프로그램이 중지됩니다.

x1, x2, ..., xn 인수를 사용하여 함수 호출

각 변수는 해당 유형의 상수, 변수 또는 식을 나타낼 수 있습니다. 전달된 인수는 쉼표로 구분되며 괄호 안에 있어야 하며, 여는 괄호는 호출된 함수의 이름 뒤에 와야 합니다.

표현식 값은 함수에 의해 반환되는 값입니다. 반환 값이 void 유형인 경우 할당 작업에서 이러한 함수 호출을 오른쪽에 배치할 수 없습니다. x1, ..., xn 식이 정확히 이 순서로 실행됩니다.

예제:

```
   int length=1000000;
   string a="a",b="b",c;
//---기타 연산
   int start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);
```

콤마 연산 (,)

쉼표로 구분된 표현식은 왼쪽에서 오른쪽으로 실행됩니다. 오른쪽 식을 계산하기 전에 왼쪽 식 계산의 모든 부작용이 나타날 수 있습니다. 결과 유형 및 값이 우측 식과 일치합니다. 전달될 매개변수 목록(위 참조)을 예로 들 수 있습니다.

예제:

```
for(i=0,j=99; i<100; i++,j--) Print(array[i][j]);
```

도트 연산자 (.)

© 2000-2025, MetaQuotes Ltd.

구조 및 클래스의 직접 공용 멤버에 대한 액세스의 경우 도트 연산이 사용됩니다. Syntax:

```
Variable_name_of_structure_type.Member_name
```

예제:

```
   struct SessionTime
     {
      string sessionName;
      int    startHour;
      int    startMinutes;
      int    endHour;
      int    endMinutes;
     } st;
   st.sessionName="Asian";
   st.startHour=0;
   st.startMinutes=0;
   st.endHour=9;
   st.endMinutes=0;
```

범위 지정 연산 (::)

mql5 프로그램의 각 기능에는 자체 실행 범위가 있습니다. 예를 들어, Print() 시스템 함수는 전역 범위에서 수행됩니다. Imported 함수는 해당 가져오기 범위 내에서 호출됩니다. classes의 메서드 함수는 해당 클래스의 범위를 가집니다. 범위 지정 연산의 구문은 다음과 같습니다:

```
[Scope_name]::Function_name(parameters)
```

범위 이름이 없는 경우 전역 범위를 사용하는 것이 명시적 방향입니다. 범위 지정 연산이 없는 경우 가장 가까운 범위에서 함수를 찾습니다. 로컬 범위에 함수가 없는 경우 전역 범위에서 검색이 수행됩니다.

범위 지정 연산은 함수 정의—클래스 멤버에도 사용됩니다.

```
type Class_name::Function_name(parameters_description)
   {
// 함수 바디
  }
```

프로그램에서 서로 다른 실행 컨텍스트에서 동일한 이름의 여러 함수를 사용하면 모호해질 수 있습니다. 명시적 범위 지정 없이 함수 호출의 우선 순위 순서는 다음과 같습니다:

1. 클래스 메서드. 클래스에 지정된 이름의 함수가 설정되어 있지 않으면 다음 수준으로 이동합니다.
2. MQL5 함수. 언어에 이러한 함수가 없는 경우 다음 단계로 이동합니다.
3. 사용자가 정의한 전역 함수. 지정한 이름의 함수를 찾을 수 없는 경우 다음 수준으로 이동합니다.
4. 임포트된 함수. 지정한 이름의 함수를 찾을 수 없으면 컴파일러에서 오류를 반환합니다.

함수 호출의 모호성을 방지하려면 항상 범위 지정 작업을 사용하여 함수 범위를 명시적으로 지정하십시오.

예제:

```
#property script_show_inputs
```

© 2000-2025, MetaQuotes Ltd.

```
#import "kernel32.dll"
   int GetLastError(void);
#import

class CCheckContext
  {
   int          m_id;
public:
                CCheckContext() { m_id=1234; }
protected:
   int          GetLastError() { return(m_id); }
  };
class CCheckContext2 : public CCheckContext
  {
   int          m_id2;
public:
                CCheckContext2() { m_id2=5678; }
   void         Print();
protected:
   int          GetLastError() { return(m_id2); }
  };
void CCheckContext2::Print()
  {
   ::Print("Terminal GetLastError",::GetLastError());
   ::Print("kernel32 GetLastError",kernel32::GetLastError());
   ::Print("parent GetLastError",CCheckContext::GetLastError());
   ::Print("our GetLastError",GetLastError());
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CCheckContext2 test;
   test.Print();
  }
//+------------------------------------------------------------------+
```

데이터 유형 개체의 크기 또는 크기(sizeof)를 가져오는 연산

sizeof 연산을 이용하여 식별자 또는 유형에 해당하는 메모리 크기를 정의할 수 있습니다. sizeof 연산은 다음과 같은 식입니다:

예제:

```
sizeof(expression)
```

대괄호로 묶인 모든 식별자 또는 유형 이름을 표현식으로 사용할 수 있습니다. void 유형 이름은 사용할 수 없으며 식별자는 비트 필드에 속하거나 함수 이름이 될 수 없습니다.

© 2000-2025, MetaQuotes Ltd.

식이 정적 배열의 이름(즉, 첫 번째 차원이 주어짐)이면 결과는 전체 배열의 크기(즉, 요소 수와 유형의 길이의 곱)가 됩니다. 표현식이 동적 배열의 이름인 경우(첫 번째 차원이 지정되지 않음) 결과는 동적 배열의 객체 크기가 됩니다.

구조 또는 클래스 유형의 이름 또는 구조 또는 클래스 유형의 식별자에 sizeof를 적용하면 결과물이 구조 또는 클래스의 실제 크기가 됩니다.

예제:

```
struct myStruct
  {
   char   h;
   int    b;
   double f;
  } str;
Print("sizeof(str) = ",sizeof(str));
Print("sizeof(myStruct) = ",sizeof(myStruct));
```

크기는 컴파일 단계에서 계산됩니다.

더 보기

우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

우선 순위 규칙

표의 각 작업 그룹에는 동일한 우선 순위가 지정됩니다. 작업의 우선 순위가 높을수록 테이블에서 그룹의 위치가 높아집니다. 우선 순위 규칙에 따라 연산자 및 피연산자 그룹이 결정됩니다.

주의: MQL5 언어의 작업 우선 순위는 C++에서 채택된 우선 순위에 해당하며 MQL4 언어로 주어진 우선 순위와 다릅니다.

연산	설명	실행 순서
() [] .	함수 호출 배열 요소 참조 구조 요소 참조	왼쪽에서 오른쪽으로
! ~ - ++ -- (type) sizeof	논리 부정 비트 부정(보완) 부호 변경 1씩 증가 1씩 감소 타입캐스팅 크기 결정(바이트)	오른쪽에서 왼쪽으로
* / %	곱하기 나누기 모듈 분할	왼쪽에서 오른쪽으로
+ -	더하기 빼기	왼쪽에서 오른쪽으로
<< >>	좌측 이동 우측 이동	왼쪽에서 오른쪽으로
< <= > >=	미만 이하 초과 이상	왼쪽에서 오른쪽으로
== !=	같음 같지않음	왼쪽에서 오른쪽으로
&	비트 AND 연산	왼쪽에서 오른쪽으로
^	비트 제외 OR	왼쪽에서 오른쪽으로
|	비트 OR 연산	왼쪽에서 오른쪽으로
&&	논리 AND 연산	왼쪽에서 오른쪽으로
||	논리 OR 연산	왼쪽에서 오른쪽으로
?:	조건부 연산자	오른쪽에서 왼쪽으로
= *= /= %=	대입 대입 포함 곱셈 대입 포함 나눗셈 대입 포함 모듈	오른쪽에서 왼쪽으로

© 2000-2025, MetaQuotes Ltd.

연산	설명	실행 순서
+= -= <<= >>= &= ^= |=	대입 포함 덧셈 대입 포함 뺄셈 대입 포함 좌측 이동 대입 포함 우측 이동 대입 포함 비트 AND 대입 제외 OR 대입 포함 비트 OR	
,	콤마	왼쪽에서 오른쪽으로

작업 실행 순서를 변경하려면 우선 순위가 높은 괄호를 사용합니다.

© 2000-2025, MetaQuotes Ltd.

연산자

언어 연산자는 작업을 수행하기 위해 실행해야 하는 일부 알고리즘 작업을 설명합니다. 프로그램 본문은 이러한 연산자의 시퀀스입니다. 연산자가 하나씩 세미콜론으로 구분됩니다.

연산자	설명
복합 연산자 {}	중괄호 {}로 묶인 모든 유형의 하나 이상의 연산자
식 연산자 (;)	세미콜론(;)으로 끝나는 식
반환 연산자	현재 기능을 종료하고 호출 프로그램으로 제어 권한을 반환합니다.
if-else 조건부 연산자	선택을 해야 할 때 사용합니다
?: 조건부 연산자	if-else 조건부 연산자의 간단한 아날로그
switch 선택 연산자	식 값에 해당하는 연산자에게 컨트롤을 전달합니다.
while 루프 연산자	선택한 표현식이 거짓이 될 때까지 연산자를 수행합니다. 각 반복 전에 식이 검사됩니다
for 루프 연산자	선택한 표현식이 거짓이 될 때까지 연산자를 수행합니다. 각 반복 전에 식이 검사됩니다
do-while 루프 연산자	선택한 표현식이 거짓이 될 때까지 연산자를 수행합니다. 각 루프 후 종료 조건이 점검됩니다. 루프 본문은 항상 한 번 이상 실행됩니다.
break 연산자	가장 가까운 외부 연산자 스위치의 실행을 종료합니다(do-while 또는 for)
continue 연산자	제어 권한을 가장 가까운 외부 루프 연산자의 시작 부분에 전달합니다(do-while 또는 for).
행렬 곱 연산자 @	@ MQL5 연산자는 선형 대수의 규칙에 따라 행렬 곱셈을 구현합니다. 벡터의 스칼라 곱셈을 수행 할 뿐만 아니라 곱하기 행렬 및 벡터 를 곱할 수 있습니다.
new 연산자	적절한 크기의 개체를 만들고 생성된 개체의 설명자를 반환합니다.
delete 연산자	새 연산자가 만든 개체를 삭제합니다

하나의 연산자가 하나 이상의 라인을 차지할 수 있습니다. 둘 이상의 연산자를 같은 줄에 배치할 수 있습니다. 실행 순서를 제어하는 연산자(만약, if-else, switching, while 및 for)를 서로 내포할 수 있습니다.

예제:

```
if(Month() == 12)
  if(Day() == 31) Print("Happy New Year!");
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

복합 연산자

복합 연산자(블록)는 중괄호 {}로 둘러싸인 모든 유형의 하나 이상의 연산자로 구성됩니다. 닫는 꺽쇠 뒤에 세미콜론(;)을 붙이면 안 됩니다.

예제:

```
if(x==0)
  {
   Print("잘못된 포지션 x = ",x);
   return;
  }
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

표현식 연산자

세미콜론(;) 뒤에 오는 식이 연산자입니다. 다음은 표현식 연산자의 몇 가지 예입니다.

대입 연산자

Identifier = expression;

```
x=3;
y=x=3;
bool equal=(x==y);
```

대입 연산자는 식에서 여러 번 사용할 수 있습니다. 이 경우 식은 오른쪽에서 왼쪽으로 처리됩니다.

함수 호출 연산자

Function_name (argument1,..., argumentN);

```
FileClose(file);
```

Empty 연산자

세미콜론(;)으로만 구성되며 컨트롤 연산자의 빈 본문을 나타내는 데 사용됩니다.

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

반환 연산자

반환 연산자는 현재 함수 실행을 종료하고 호출 프로그램에 대한 제어를 반환합니다. 표현식 계산 결과가 호출 함수로 반환됩니다. 표현식은 할당 연산자를 포함할 수 있습니다.

예제:

```
int CalcSum(int x, int y)
  {
   return(x+y);
  }
```

void 반환 유형이 있는 함수에서는 표현식이 없는 return 연산자를 사용해야 합니다:

```
void SomeFunction()
  {
   Print("Hello!");
   return;    // 이 연산자를 제거할 수 있습니다
  }
```

함수의 오른쪽 중괄호는 표현식 없이 반환 연산자를 암시적으로 실행함을 의미합니다.

반환 가능 항목: 단순 유형, 단순 구조, 객체 포인터. *반환* 연산자를 사용하면 배열, 클래스 객체, 복합 구조 유형의 변수를 반환할 수 없습니다.

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

If-Else 조건부 연산자

IF -ELSE 연산자는 선택이 필수적일 경우 사용됩니다. 공식적으로 구문은 다음과 같습니다:

```
if (expression)
    연산자1
else
    연산자2
```

표현식이 true이면 연산자 1이 실행되고 연산자 2 뒤에 오는 연산자(연산자 2는 실행되지 않음)가 제어됩니다. 표현식이 false이면 연산자 2가 실행됩니다.

else 파트(if 연산자의)는 생략될 수 있습니다. 따라서, 연산에 발산이 나타날 수 있습니다, if 연산자가 else 파트를 생략한 경우. 이 경우, else 부분이 없는 동일한 블록에서 가장 가까운 이전 if 연산자에게 else 주소를 지정합니다.

예제:

```
//--- else 부분은 연산자가 두 번째 파트를 참조합니다:
if(x>1)
   if(y==2) z=5;
else     z=6;
//--- else 부분은 첫 번째 if 연산자를 참조합니다:
if(x>1)
  {
   if(y==2) z=5;
  }
else        z=6;
//--- 내포 연산자
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("ERROR");
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

3진 연산자 ?:

3차 연산자의 일반적인 형태는 다음과 같습니다:

```
표현식1 ? 표현식2 : 표현식3
```

첫 번째 피연산자 -"표현식1" -bool 유형 값이 되는 표현식을 사용할 수 있습니다. 결과가 true이면, 두 번째 피연산자에 의해 설정된 연산자, 즉 "표현식2"가 실행됩니다.

첫 번째 피연산자가 false이면, 세 번째 피연산자 -"표현식3"가 수행됩니다. 두 번째와 세 번째 피연산자, 즉 "식2"와 "식3"는 한 유형의 값을 반환해야 하며 void 유형이어서는 안됩니다. 조건부 연산자 실행의 결과는 표현식1의 결과에 따라 표현식2의 결과 또는 표현식3의 결과입니다.

```
//--- 일일 시가와 종가의 차이를 정상화
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

이 항목은 다음과 같습니다:

```
   double true_range;
   if(High==Low)true_range=0;                // 고가와 저가가 동일한 경우
   else true_range=(Close-Open)/(High-Low); // if the range is not null
```

연산자 사용 제한

연산자는 "표현식1" 값을 기반으로 "표현식2" 또는 "표현식3" 중 하나를 반환해야 합니다. 이러한 표현식에는 몇 가지 제한이 있습니다:

1. 사용자 정의 유형과 단순 유형 또는 열거형을 함께 사용하지 마십시오. NULL을 pointer에 사용할 수 있습니다.
2. 값 유형이 단순할 경우 연산자는 최대 유형이 됩니다(Type casting 참조).
3. 값 중 하나가 열거형이고 두 번째 값이 숫자형이면 열거형이 int로 대체되고 두 번째 규칙이 적용됩니다.
4. 두 값이 모두 열거형이면 유형이 동일해야 하며 연산자는 열거형입니다.

사용자 정의 유형 제한(클래스 또는 구조):

a) 유형은 동일해야 하며 그렇지 않으면 다른 유형에서 파생되어야 합니다.

b) 유형이 동일하지 않으면(상속) 자식 항목이 부모 유형으로 암시적으로 캐스팅됩니다. 즉, 연산자가 부모 유형이 됩니다.

c) 개체와 포인터를 혼합하지 마십시오. 두 식 모두 개체 또는 포인터여야 합니다. NULL은 포인터용으로 쓰일 수 있습니다.

참고

조건부 연산자를 오버로드된 함수의 인수로 사용할 때는 프로그램 컴파일 시 조건부 연산자의 결과 유형이 정의되므로 주의하십시오. 그리고 이 유형은 "표현식2"와 "표현식3" 중 큰 유형으로 결정됩니다.

예제:

```
void func(double d) { Print("double argument: ",d); }
void func(string s) { Print("string argument: ",s); }
```

© 2000-2025, MetaQuotes Ltd.

```
bool   Expression1=true;
double Expression2=M_PI;
string Expression3="3.1415926";

void OnStart()
  {
   func(Expression2);
   func(Expression3);

   func(Expression1?Expression2:Expression3);   // 문자열에 암시적 캐스팅에 대한 경고
   func(!Expression1?Expression2:Expression3);  // 문자열에 암시적 캐스팅에 대한 경고
  }

//   Result:
//   double argument: 3.141592653589793
//   string argument: 3.1415926
//   string argument: 3.141592653589793
//   string argument: 3.1415926
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

スイッチ演算子

表現式 値を モ든 *ケイス* 変数の 상수와 비교하고 표현식 값에 해당하는 연산자에 컨트롤을 전달합니다. *ケイス*의 각 변수는 정수형 상수, 리터럴 상수 또는 상수 식을 사용하여 표시할 수 있습니다. 상수 식에 변수 또는 함수 호출을 포함할 수 없습니다. *スイッチ* 연산자의 표현식은 정수 형식(int 또는 unt)이어야 합니다.

```
switch(expression)
  {
   case constant: operators
   case constant: operators
      ...
   default: operators
  }
```

다음과 같은 경우 ***default*** 레이블로 표시된 연산자가 실행됩니다: ***case*** 연산자에 표현식 값과 같은 상수가 없는 경우. ***default*** 변수를 반드시 선언해서는 안 되며 반드시 마지막 변수가 되어서도 안 됩니다. 표현식 값에 해당하는 상수가 없고 ***default*** 변수를 사용할 수 없는 경우 작업이 실행되지 않습니다.

상수가 포함된 ***case*** 키워드는 레이블일 뿐이며, 일부 ***case*** 변수에 대해 연산자가 실행되는 경우, 프로그램은 ***break*** 연산자가 발생할 때까지 이후의 모든 변형 연산자를 추가로 실행합니다. 여러 변종으로 연산자 시퀀스를 바인딩할 수 있습니다.

컴파일하는 동안 상수 식이 계산됩니다. 하나의 *スイッチ* 연산자에 있는 두 상수는 동일한 값을 가질 수 없습니다.

예제:

```
//--- 첫 번째 예제
switch(x)
  {
   case 'A':
      Print("CASE A");
      break;
   case 'B':
   case 'C':
      Print("CASE B or C");
      break;
   default:
      Print("NOT A, B or C");
      break;
  }

//---  두번째 예제
   string res="";
   int i=0;
   switch(i)
     {
      case 1:
         res=i;break;
      default:
```

© 2000-2025, MetaQuotes Ltd.

```
         res="default";break;
      case 2:
         res=i;break;
      case 3:
         res=i;break;
    }
   Print(res);
/*
   Result
   default
*/
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

While 루프 연산자

while 연산자는 선택된 식과 연산자로 구성되며, 반드시 이행되어야 합니다:

```
while(expression)
 연산자;
```

표현식이 참이면 표현식이 거짓이 될 때까지 연산자가 실행됩니다. 표현식이 false이면 컨트롤이 다음 연산자에게 전달됩니다. 연산자가 실행되기 전에 식 값이 정의됩니다. 따라서 식이 처음부터 false이면 연산자가 전혀 실행되지 않습니다.

참고

많은 반복이 루프에서 처리될 것으로 예상되는 경우 IsStopped() 함수를 사용하여 강제 프로그램 종료 사실을 확인하는 것이 좋습니다.

예제:

```
while(k<n && !IsStopped())
  {
   y=y*x;
   k++;
  }
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

For 루프 연산자

for 연산자는 세 표현식과 실행 연산자로 구성됩니다.

```
for(expression1; expression2; expression3)
  연산자;
```

표현식1은 루프 초기화를 설명합니다. 표현식2는 루프 종료의 조건을 확인합니다. true라면, 루프의 for 바디는 실행됩니다. 루프는 false가 될 때까지 표현식2를 반복합니다. false이면 루프가 종료되고 다음 연산자에게 제어가 제공됩니다. 표현식 3은 각 반복 후에 계산됩니다.

for 연산자는 다음과 같은 일련의 연산자와 동일합니다:

```
표현식1;
while(표현식2)
  {
  연산자;
  표현식3;
  };
```

for 연산자에 세 식 중 하나 또는 세 식이 모두 없을 수 있지만 이를 구분하는 세미콜론(;)을 생략해서는 안 됩니다. 표현식 2를 생략하면 항상 참인 것으로 간주됩니다. for(;;) 연산자는 연속적 루프이며, while(1) 연산자와 동일합니다. 각 표현식 1 또는 3은 쉼표 연산자 ','에 의해 결합된 여러 표현식으로 구성될 수 있습니다.

참고

많은 반복이 루프에서 처리될 것으로 예상되는 경우 IsStopped() 함수를 사용하여 강제 프로그램 종료 사실을 확인하는 것이 좋습니다.

예제:

```
for(x=1;x<=7000; x++)
  {
   if(IsStopped())
      break;
   Print(MathPower(x,2));
  }
//--- 다른 예
for(;!IsStopped();)
  {
   Print(MathPower(x,2));
   x++;
   if(x>10) break;
  }
//--- 세 번째 예
for(i=0,j=n-1;i<n && !IsStopped();i++,j--) a[i]=a[j];
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

루프 연산자 do while

for 및 while 루프는 종료를 루프 끝에서가 아니라 처음부터 확인합니다. 세 번째 루프 연산자 do - while은 각 루프 반복 후 끝에서 종료 상태를 확인합니다. 루프의 바디는 항상 한 번 이상 실행됩니다.

```
do
   연산자;
while(expression);
```

연산자가 먼저 실행된 다음 식이 계산됩니다. true이면 연산자가 다시 실행되는 등의 작업을 수행합니다. 식이 false이면 루프가 종료됩니다.

참고

많은 반복이 루프에서 처리될 것으로 예상되는 경우 IsStopped() 함수를 사용하여 강제 프로그램 종료 사실을 확인하는 것이 좋습니다.

예제:

```
//--- 피보나치 수열 연산
   int counterFibonacci=15;
   int i=0,first=0,second=1;
   int currentFibonacciNumber;
   do
     {
      currentFibonacciNumber=first+second;
      Print("i = ",i,"  currentFibonacciNumber = ",currentFibonacciNumber);
      first=second;
      second=currentFibonacciNumber;
      i++; // 이 연산자가 없으면 무한 루프가 나타납니다!
     }
   while(i<counterFibonacci && !IsStopped());
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

중단 연산자

중단 연산자는 가장 가까운 중첩된 외부 switch, while, do-while 또는 for 연산자의 실행을 종료합니다. 제어는 종료된 연산자 다음에 오는 연산자에게 전달됩니다. 이 연산자의 목적 중 하나는 특정 값이 변수에 할당될 때 루핑 실행을 완료하는 것입니다.

예제:

```
//--- 첫 번째 0 요소 검색
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

계속 연산자

계속 연산자는 컨트롤을 가장 가까운 외부 루프 while, do-while 또는 for 연산자의 시작에 전달하고 다음 반복이 호출됩니다. 이 연산자의 목적은 중단 연산자의 목적과 반대입니다.

예제:

```
//--- 0이 아닌 모든 요소의 합계
int func(int array[])
  {
   int array_size=ArraySize(array);
   int sum=0;
   for(int i=0;i<array_size; i++)
     {
      if(a[i]==0) continue;
      sum+=a[i];
    }
   return(sum);
  }
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

행렬 곱 연산자

@ MQL5 연산자는 선형 대수의 규칙에 따라 행렬 곱셈을 구현합니다. 벡터의 스칼라 곱셈을 수행 할 뿐만 아니라 곱하기 행렬 및 벡터 를 곱할 수 있습니다.

지원되는 요소 타입:

- float
- double
- complex<float>
- complex<double>

중요 : 왼쪽 및 오른쪽 피연산자의 요소 유형이 일치해야합니다. .

사용 예

1. 행렬 곱하기 (행렬 ×행렬)

```
matrix A(2, 3);
matrix B(3, 2);
matrix C = A @ B; // 결과: 사이즈 [2,2]인 행렬 C
```

2. 행렬 곱하기(행렬 ×벡터)

```
matrix M(2, 3);
vector V(3);
vector R = M @ V; // 결과: 2 요소인 벡터 R
```

3. 행렬 곱하기 (벡터 x 행렬)

```
matrix M(2, 3);
vector V(1, 2);
vector R = V @ M; // 결과: 3 요소인 벡터 R
```

4. 스칼라 곱하기 (벡터 ×벡터)

```
vector V1(1, 3), V2(1, 3);
double r = V1 @ V2; // 결과: 스칼라
```

주의

차원은 곱셈 규칙을 준수해야합니다: 첫 번째 피연산자 열 수 = 두 번째 행의 수.

차원 오류의 경우 런타임에 예외가 트리거됩니다.

왼쪽의 벡터는 수평으로 간주됩니다 (1 ×N).

© 2000-2025, MetaQuotes Ltd.

오른쪽의 벡터는 수직으로 간주됩니다 (n × 1).

@연산의 우선 순위는 곱셈의 우선 순위, 즉 d = c+a@b의 입력에 해당합니다. 첫 번째 행렬 곱셈 t = a@b가 수행되고 요소 별 추가 d = c+t가 수행됩니다.

더 참조할 만한 내용

MatMul, GeMM, Kron, Dot,

© 2000-2025, MetaQuotes Ltd.

객체 생성 연산자 new

new 연산자는 자동으로 해당 크기의 객체를 만들고 객체 생성자를 호출한 후 생성된 객체의 설명자를 반환합니다. 오류가 발생하면 연산자는 NULL 상수와 비교할 수 있는 null 설명자를 반환합니다.

new 연산자는 클래스 객체에만 적용할 수 있습니다. 구조에는 적용할 수 없습니다

연산자를 사용하여 객체 배열을 작성해서는 안 됩니다. 이것을 하기 위해선, ArrayResize() 함수를 사용하십시오.

예제:

```
//+------------------------------------------------------------------+
//| Figure 생성                                                      |
//+------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- 가능한 7가지 도형 중 하나를 임의로 만듭니다
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- 그리기
   if(m_shape!=NULL)
     {
      //--- pre-settings
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- 그리기
      m_shape.Draw();
     }
//---
  }
```

개체 설명자는 메모리 주소에 대한 포인터가 아닙니다.

새 연산자로 만든 개체는 삭제 연산자를 사용하여 명시적으로 제거해야 합니다.

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

객체 삭제 연산자 삭제

삭제 연산자는 new 연산자가 만든 객체를 삭제하고 해당 클래스 소멸자를 호출하며 객체가 점유한 메모리를 비웁니다. 기존 객체의 실제 설명자가 피연산자로 사용됩니다. 삭제 연산이 실행되면 객체 설명자가 유효하지 않게 됩니다.

예제:

```
      //--- 삭제 모형
      delete m_shape;
      m_shape=NULL;
      //--- create a new figure
      NewShape();
```

더 보기

변수 초기화, 가시성 범위 및 변수 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

함수

모든 태스크는 하위 태스크로 나눌 수 있으며, 각 태스크는 코드 형식으로 직접 표시하거나 더 작은 하위 태스크로 나눌 수 있습니다. 이 방법을 *단계별 세분화*라고 합니다. 기능은 해결할 하위 작업의 코드를 작성하는 데 사용됩니다. 함수의 기능을 설명하는 코드를 다음과 같이 부릅니다, *함수 정의:*

```
function_header
  {
   instructions
  }
```

첫 번째 꺽쇠 앞에 있는 것은 함수 정의의 ***header***이며, 꺽쇠 사이에 있는 것은 함수 정의의 ***body***입니다. 함수 헤더에는 반환 값 유형, 이름(식별자) 및 형식 매개변수에 대한 설명이 포함됩니다. 함수에 전달되는 매개 변수 수는 제한되며 64개를 초과할 수 없습니다.

이 기능은 프로그램의 다른 부분에서 필요한 횟수만큼 호출할 수 있습니다. 실제로 반환 유형, 함수 식별자 및 매개변수 유형은 *함수 프로토타입*을 구성합니다.

함수 프로토타입은 함수 선언이지만 정의는 아닙니다. 반환 유형의 명시적 선언과 인수 유형의 목록으로 인해 함수 호출 중에 엄격한 유형 확인 및 암시적 유형 캐스팅이 가능합니다. 코드 가독성을 높이기 위해 함수 선언이 클래스에 사용되는 경우가 매우 많습니다.

함수 정의는 해당 선언과 정확히 일치해야 합니다. 선언된 각 함수를 정의해야 합니다.

예제:

```
double                          // 값 유형 반환
linfunc (double a, double b) // 함수 이름 및 매개 변수 목록
  {
                                // 복합 연산자
   return (a + b);              // 값 반환
  }
```

return연산자는 이 연산자에 있는 표현식의 값을 반환할 수 있습니다. 필요한 경우 표현식 값이 함수 결과 유형으로 변환됩니다. 반환 가능 항목: simple types, simple structures, object pointers. *return* 연산자를 사용하면 배열, 클래스 객체, 복합 구조 유형의 변수를 반환할 수 없습니다.

값을 반환하지 않는 함수는 void 유형의 함수로 서술되어야 합니다.

예제:

```
void errmesg(string s)
  {
   Print("error: "+s);
  }
```

함수에 전달되는 매개 변수는 해당 유형의 상수로 정의된 기본값을 가질 수 있습니다.

예제:

```
int somefunc(double a,
```

© 2000-2025, MetaQuotes Ltd.

```
              double d=0.0001,
              int n=5,
              bool b=true,
              string s="전달된 문자열")
  {
   Print("필수 파라미터 a = ",a);
   Print("다음 매개 변수 전달: d = ",d," n = ",n," b = ",b," s = ",s);
   return(0);
  }
```

매개변수에 기본값이 있는 경우 이후의 모든 매개변수에도 기본값이 있어야 합니다.

잘못된 선언의 예:

```
int somefunc(double a,
              double d=0.0001,    // 기본값 0.0001 선언됨
              int n,              // 기본값이 지정되지 않았습니다 !
              bool b,             // 기본값이 지정되지 않았습니다 !
              string s="전달된 문자열")
  {
  }
```

더 보기

오버로드, 가상 함수, 다형성

© 2000-2025, MetaQuotes Ltd.

함수 호출

이전에 설명되지 않은 이름이 표현식에 나타나고 왼쪽 괄호 뒤에 오는 경우 상황에 따라 함수의 이름으로 간주됩니다.

```
function_name (x1, x2,..., xn)
```

인수(공식 매개변수)는 값별로 전달됩니다, 즉, 각 표현식 x1,..., xn이 계산되고 값이 함수에 전달됩니다. 수식 계산 순서와 값 로드 순서는 보장되지 않습니다. 실행 중에는 시스템은 함수에 전달된 인수 수와 유형을 확인합니다. 이러한 함수의 주소 지정 방식을 값 호출이라고 합니다.

함수 호출은 수식이며, 이 값은 함수에 의해 반환되는 값입니다. 위에서 설명한 함수 유형은 반환 값의 유형과 일치해야 합니다. 이 함수는 글로벌 범위에 있는 프로그램의 모든 부분, 즉 다른 함수에서 선언되거나 설명될 수 있습니다. 다른 함수 내부에서 함수를 선언하거나 설명할 수 없습니다.

예제:

```
int start()
  {
   double some_array[4]={0.3, 1.4, 2.5, 3.6};
   double a=linfunc(some_array, 10.5, 8);
   //...
  }
double linfunc(double x[], double a, double b)
  {
   return (a*x[0] + b);
  }
```

기본 매개 변수를 사용하여 함수를 호출할 때 전달될 매개 변수 목록을 제한할 수 있지만 첫 번째 기본 매개 변수 앞은 사용할 수 없습니다.

예제:

```
void somefunc(double init,
              double sec=0.0001, //기본 값 설정
              int level=10);
//...
somefunc();                       // 잘못된 호출. 첫 번째 매개 변수를 제공해야 합니다.
somefunc(3.14);                   // 정상적 호출
somefunc(3.14,0.0002);            // 정상적 호출
somefunc(3.14,0.0002,10);         // 정상적 호출
```

함수를 호출할 때 다음과 같은 기본값을 가진 매개 변수를 건너뛸 수 없습니다.

```
somefunc(3.14, , 10);             // 잘못된 호출 -> 두 번째 매개 변수를 건너뛰었습니다.
```

더 보기

오버로드, 가상 함수, 다형성

© 2000-2025, MetaQuotes Ltd.

전달 매개 변수

기계어에는 하위 프로그램(함수)에 인수를 전달할 수 있는 두 가지 방법이 있습니다. 첫 번째 방법은 값별로 파라미터를 보내는 것입니다. 이 방법은 인수 값을 형식 함수 매개 변수로 복사합니다. 그러므로, 함수 내에서 이 매개변수의 변경은 해당 호출 인수에 영향을 미치지 않습니다.

```
//+------------------------------------------------------------------+
//| 값별로 매개 변수 전달                                            |
//+------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("호출 전 a와 b:",a," ",b);
   double d=FirstMethod(a,b);
   Print("호출 후 a와 b:",a," ",b);
  }
//--- 스크립트 실행 결과
//  호출 전 a와 b:  14 8
//  호출 후 a와 b:  14 8
```

두 번째 방법은 레퍼런스로 통과시키는 것입니다. 이 경우 파라미터(값이 아님)에 대한 참조가 함수 파라미터로 전달됩니다. 함수 내부에서는 호출에 지정된 실제 파라미터를 나타내는 데 사용됩니다. 이는 매개변수 변경사항이 함수를 호출하는 데 사용되는 인수에 영향을 미친다는 것을 의미합니다.

```
//+------------------------------------------------------------------+
//| 레퍼런스를 통해 매개 변수 전달                                   |
//+------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("호출 전 a와 b:",a," ",b);
   double d=SecondMethod(a,b);
   Print("호출 후 a와 b:",a," ",b);
  }
//+------------------------------------------------------------------+
//--- 스크립트 실행 결과
//  호출 전 a와 b: 14 8
//  호출 후 a와 b: 28 4
```

MQL5는 두 가지 방법을 모두 사용합니다. 단, 배열, 구조 유형 변수 및 클래스 개체는 항상 참조를 통해 전달됩니다. 실제 매개변수(함수 호출 시 전달되는 인수)의 변경을 방지하려면 액세스 지정자 const를 사용합니다. *const* 지정자로 선언된 변수의 내용을 변경하려고 하면 컴파일러에서 오류가 발생합니다.

참고

매개변수는 역순으로 함수에 전달됩니다. 즉, 먼저 마지막 매개변수를 계산하여 전달한 다음 마지막 매개변수를 제외한 나머지 매개변수가 전달됩니다. 마지막으로 계산하여 통과한 파라미터는 괄호를 열고 맨 앞에 있는 파라미터입니다.

예제:

```
void OnStart()
  {
//---
   int a[]={0,1,2};
   int i=0;

   func(a[i],a[i++],"First call (i = "+string(i)+")");
   func(a[i++],a[i],"Second call (i = "+string(i)+")");
// Result:
// First call (i = 0) : par1 = 1     par2 = 0
// Second call (i = 1) : par1 = 1     par2 = 1

  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void func(int par1,int par2,string comment)
  {
   Print(comment,": par1 = ",par1,"    par2 = ",par2);
  }
```

© 2000-2025, MetaQuotes Ltd.

첫 번째 호출에서(위의 예 참조) ***i*** 변수는 문자열 연결에 처음 사용됩니다:

```
"First call (i = "+string(i)+")"
```

여기서 그것의 값은 변하지 않습니다. 그런 다음 ***i*** 변수가 ***a[i++]*** 배열 요소를 계산할 때 사용되는데, 즉, 인덱스 i가 있는 배열 요소에 액세스할 때, ***i*** 변수는 증가됩니다. 그리고 그 후에만 ***i*** 변수의 값이 변경된 첫 번째 파라미터가 계산됩니다.

두 번째 호출에서는 세 개의 파라미터를 모두 계산할 때 동일한 값 i(함수 호출의 첫 번째 페이즈에서 계산됨)가 사용됩니다. 첫 번째 파라미터를 계산한 후에만 ***i*** 변수가 다시 변경됩니다.

더 보기

가시성 범위 및 변수의 수명, 오버로드, 가상 함수, 다형성

© 2000-2025, MetaQuotes Ltd.

함수 오버로드

일반적으로 함수 이름은 주요 용도를 반영하는 경향이 있습니다. 일반적으로 읽을 수 있는 프로그램에는 잘 선택된 다양한 식별자가 포함됩니다. 때로는 동일한 목적으로 서로 다른 함수가 사용될 수 있습니다. 예를 들어, 정밀도가 2배인 숫자와 같은 함수의 평균 값을 계산하지만 정수 배열을 사용하여 연산하는 함수를 고려해 보겠습니다. 둘 다 AverageFromArray라고 부르기에 편리합니다:

```
//+------------------------------------------------------------------+
//| double 유형의 배열에 대한 평균 계산                                |
//+------------------------------------------------------------------+
double AverageFromArray(const double & array[],int size)
  {
   if(size<=0) return 0.0;
   double sum=0.0;
   double aver;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];    // double에 대한 합
     }
   aver=sum/size;       // 합계를 숫자로 나눔
//---
   Print("double 유형의 배열에 대한 평균 계산" );
   return aver;
  }
//+------------------------------------------------------------------+
//| int 유형의 배열에 대한 평균 계산                                  |
//+------------------------------------------------------------------+
double AverageFromArray(const int & array[],int size)
  {
   if(size<=0) return 0.0;
   double aver=0.0;
   int sum=0;
//---
   for(int i=0;i<size;i++)
     {
      sum+=array[i];     // int에 대한 합
     }
   aver=(double)sum/size;// double 타입의 양을 주고 나눕니다
//---
   Print("int 유형의 배열에 대한 평균 계산");
   return aver;
  }
```

각 함수에는 Print() 함수를 통한 메시지 출력이 포함됩니다;

```
   Print("int 유형의 배열에 대한 평균 계산");
```

컴파일러는 인수의 유형과 인수의 수량에 따라 필요한 함수를 선택합니다. 선택이 이루어지는 규칙을 *서명 매칭 알고리즘*이라고 합니다. 서명은 함수 선언에 사용되는 유형의 목록입니다.

© 2000-2025, MetaQuotes Ltd.

예제:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int    a[5]={1,2,3,4,5};
   double b[5]={1.1,2.2,3.3,4.4,5.5};
   double int_aver=AverageFromArray(a,5);
   double double_aver=AverageFromArray(b,5);
   Print("int_aver = ",int_aver,"   double_aver = ",double_aver);
  }
//--- 스크립트 결과
// int 유형의 배열에 대한 평균 계산
// double 유형의 배열에 대한 평균 계산
// int_aver= 3.00000000    double_aver= 3.30000000
```

함수 오버로딩은 이름은 같지만 매개 변수는 다른 여러 함수를 만드는 프로세스입니다. 이는 오버로드된 함수 변형에서 인수 및/또는 인수 유형이 달라야 함을 의미합니다. 함수를 호출할 때 인수 목록과 함수 선언의 매개 변수 목록에 대한 대응 관계를 기반으로 특정 함수 변형이 선택됩니다.

오버로드된 함수를 호출할 때 컴파일러에는 적절한 함수를 선택하는 알고리즘이 있어야 합니다. 이 선택을 수행하는 알고리즘은 존재하는 유형의 캐스팅에 따라 달라집니다. 최고의 통신은 고유해야 합니다. 오버로드된 함수는 하나 이상의 인수에 대해 다른 모든 변형 중 가장 적합해야 합니다. 동시에 다른 변수가 아닌 모든 인수와 일치해야 합니다.

아래는 각 인수에 대한 일치 알고리즘입니다.

오버로드 함수 선택 알고리즘

1. 가능한 경우 엄격 매칭을 사용합니다.
2. 표준 유형 증가를 시도합니다.
3. 표준 타입 캐스팅을 시도합니다.

다른 표준 변환보다 표준 유형 증가가 좋습니다. 증가는 float을 double로, bool, char, short 또는 enum을 int로 변환하는 것입니다. 유사한 정수형 배열의 타입캐스팅도 타입캐스팅에 속합니다. 유사한 유형은 다음과 같습니다: bool, char, uchar 세 유형이 모두 싱글 바이트 정수이므로; double 바이트 정수 short와 ushort; 4바이트 정수 int, unit 및 color; long, ulong 및 datetime.

물론 엄격 매칭이 최고입니다. 이러한 일관성을 얻으려면 타입캐스팅을 사용할 수 있습니다. 컴파일러는 애매한 상황에 대처할 수 없습니다. 따라서 오버로드된 함수를 불명확하게 만드는 유형의 미묘한 차이와 암시적 변환에 의존해서는 안 됩니다.

의심스러운 경우 명시적 변환을 사용하여 엄격한 규정 준수를 보장하십시오.

MQL5에서 오버로드된 함수의 예는 ArrayInitialize() 함수의 예에서 볼 수 있습니다.

함수 오버로딩 규칙은 class method의 오버로드에 적용됩니다.

© 2000-2025, MetaQuotes Ltd.

시스템 기능의 오버로딩은 허용되지만 컴파일러는 필요한 기능을 정확하게 선택할 수 있는지 확인해야 합니다. 예를 들어 시스템 함수 MathMax()를 4가지 방법으로 오버로드할 수 있지만 두 가지 변형만 맞습니다.

예제:

```
// 1. 오버로드 허용 - 함수가 파라미터 수에서 내장 MathMax() 함수와 다릅니다
double MathMax(double a,double b,double c);

// 2. 오버로드가 허용되지 않습니다!
// 매개 변수 수는 다르지만 마지막 값이 기본값입니다
// 이는 호출 시 시스템 기능을 숨기는 결과로 이어지며, 이는 허용되지 않습니다
double MathMax(double a,double b,double c=DBL_MIN);

// 3. 오버로드 허용 - 매개 변수 a 및 b 유형별 일반 오버로드
double MathMax(int a,int b);

// 4. 오버로드가 허용되지 않습니다!
// 파라미터의 수와 유형이 오리지날 double MathMax(double a, double b)와 동일합니다.
int MathMax(double a,double b);
```

더 보기

오버로드, 가상 함수, 다형성

© 2000-2025, MetaQuotes Ltd.

작업 오버로드

코드를 쉽게 읽고 쓸 수 있도록 일부 작업의 오버로드가 허용됩니다. 오버로드 연산자는 operator 키워드를 사용하여 작성됩니다. 다음 연산자가 오버로드될 수 있습니다:

- binary +,-,/,*,%,<<,>>,==,!=,<,>,<=,>=,=,+=,-=,/=,*=,%=,&=,|=,^=,<<=,>>=,&&,||,&,|,^
- unary +,-,++,--,!,~
- 대입 연산자 =
- 인덱싱 연산자 []

연산 오버로드를 사용하면 복잡한 객체(구조 및 클래스)에 대해 연산 표기법(단순식 형식으로 작성됨)을 사용할 수 있습니다. 오버로드된 작업을 사용하여 식을 작성하면 더 복잡한 구현이 숨겨지기 때문에 소스 코드의 보기가 간소화됩니다.

예를 들어, 실수 부분과 가상 부분으로 구성된 복소수를 생각해 보십시오. 그것들은 수학에서 널리 사용됩니다. MQL5 언어에는 복소수를 나타내는 데이터 유형이 없지만 구조 또는 클래스 형식으로 새 데이터 유형을 만들 수 있습니다. 복합 구조를 선언하고 네 가지 산술 연산을 구현하는 네 가지 방법을 정의합니다:

```
//+------------------------------------------------------------------+
//| 복소수 연산을 위한 구조                                          |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // 실수 부분
   double            im; // 허수 부분
   //--- 생성자
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- 산술 연산
   complex           Add(const complex &l,const complex &r) const;  // 덧셈
   complex           Sub(const complex &l,const complex &r) const;  // 뺄셈
   complex           Mul(const complex &l,const complex &r) const;  // 곱셈
   complex           Div(const complex &l,const complex &r) const;  // 나눗셈
  };
```

이제 코드에서 복소수를 나타내는 변수를 선언하고 함께 작업할 수 있습니다.

예:

```
void OnStart()
  {
//--- 복합 유형의 변수 선언 및 초기화
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- 두 숫자의 합
   complex z;
```

© 2000-2025, MetaQuotes Ltd.

```
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- 두 숫자의 곱
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- 두 숫자의 나누기
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

그러나 복소수가 있는 일반 산술 연산에는 일반 연산자 "+", "-", "*" 및 "/"를 사용하는 것이 더 편리합니다.

키워드 연산자는 유형 변환을 수행하는 멤버 함수를 정의하는 데 사용됩니다. 클래스 개체 변수에 대한 단항 및 이진 작업은 정적 멤버가 아닌 함수로 오버로드될 수 있습니다. 클래스 객체에 암시적으로 작용합니다.

대부분의 이진 작업은 클래스 변수 또는 이 클래스의 개체에 대한 포인터로 하나 또는 둘 모두를 사용하는 일반 함수처럼 오버로드될 수 있습니다. 복소수 유형의 경우, 선언문에 오버로드가 발생하면 다음과 같습니다:

```
   //--- 연산자
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
```

스크립트의 전체 예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 복합형 변수 선언 및 초기화
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
   //a.re=5;
   //a.im=1;
   //b.re=-1;
   //b.im=-5;
//--- 두 숫자의 합
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- 두 숫자의 곱

   z=a*b;
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- 두 숫자의 나누기
   z=a/b;
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------+
//| 복소수 연산을 위한 구조                                          |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // 실수 부분
   double            im; // 허수 부분
   //--- 생성자
                     complex():re(0.0),im(0.0) {  }
                     complex(const double r):re(r),im(0.0) {  }
                     complex(const double r,const double i):re(r),im(i) {  }
                     complex(const complex &o):re(o.re),im(o.im) { }
   //--- 산술 연산
   complex           Add(const complex &l,const complex &r) const;  // 덧셈
   complex           Sub(const complex &l,const complex &r) const;  // 뺄셈
   complex           Mul(const complex &l,const complex &r) const;  // 곱셈
   complex           Div(const complex &l,const complex &r) const;  // 나눗셈
   //--- 이진 연산자
   complex operator+(const complex &r) const { return(Add(this,r)); }
   complex operator-(const complex &r) const { return(Sub(this,r)); }
   complex operator*(const complex &r) const { return(Mul(this,r)); }
   complex operator/(const complex &r) const { return(Div(this,r)); }
  };
//+------------------------------------------------------------------+
//| 더하기                                                           |
//+------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//--- 결과
   return res;
  }
//+------------------------------------------------------------------+
//| 빼기                                                             |
//+------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re-r.re;
   res.im=l.im-r.im;
//--- 결과
   return res;
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 곱하기                                                           |
//+------------------------------------------------------------------+
complex complex::Mul(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re*r.re-l.im*r.im;
   res.im=l.re*r.im+l.im*r.re;
//--- 결과
   return res;
  }
//+------------------------------------------------------------------+
//| 나누기                                                           |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- 빈 복소수
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- 0 확인
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+": 이 값은 0입니다");
      return(res);
     }
//--- 보조 변수
   double e;
   double f;
//--- 연산 변수 선택
   if(MathAbs(r.im)<MathAbs(r.re))
     {
      e = r.im/r.re;
      f = r.re+r.im*e;
      res.re=(l.re+l.im*e)/f;
      res.im=(l.im-l.re*e)/f;
     }
   else
     {
      e = r.re/r.im;
      f = r.im+r.re*e;
      res.re=(l.im+l.re*e)/f;
      res.im=(-l.re+l.im*e)/f;
     }
//--- 결과
   return res;
  }
```

© 2000-2025, MetaQuotes Ltd.

클래스에 대한 대부분의 단항 연산은 단일 클래스 개체 인수 또는 해당 클래스에 대한 포인터를 허용하는 일반 함수로 오버로드될 수 있습니다. 단항 작업 "-및 "!"의 오버로드를 추가합니다.

```
//+------------------------------------------------------------------+
//| 복소수 연산을 위한 구조                                          |
//+------------------------------------------------------------------+
struct complex
  {
   double            re;       // 실수 부분
   double            im;       // 허수 부분
...
   //--- 단항 연산자
   complex operator-()  const; // 단항 마이너스
   bool    operator!()  const; // 부정
  };
...
//+------------------------------------------------------------------+
//| "단항 마이너스" 연산자 오버로드                                  |
//+------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- 결과
   return res;
  }
//+------------------------------------------------------------------+
//| "논리적 부정" 연산자 오버로드                                    |
//+------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- 복소수의 실수 부분과 허수 부분이 0입니까?
   return (re!=0 && im!=0);
  }
```

이제 복소수 값을 0으로 확인하고 음수 값을 얻을 수 있습니다:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 복합형 변수 선언 및 초기화
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- 두 숫자를 나눕니다
```

© 2000-2025, MetaQuotes Ltd.

```
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- 복소수는 기본적으로 0입니다(기본 생성자 re==0 및 im==0).
   complex zero;
   Print("!zero=",!zero);
//--- 음수 값 할당
   zero=-z;
   PrintFormat("z=%.2f+i*%.2f,  zero=%.2f+i*%.2f",z.re,z.im, zero.re,zero.im);
   PrintFormat("-zero=%.2f+i*%.2f",-zero.re,-zero.im);
//--- 0을 다시 한 번 확인합니다
   Print("!zero=",!zero);
//---
  }
```

단순 유형 구조는 서로 직접 복사할 수 있으므로 대입 연산자 "="에 오버로드가 걸리지 않아도 됩니다. 따라서, 우리는 이제 일반적인 방법으로 복소수와 관련된 계산을 위한 코드를 작성할 수 있습니다.

인덱싱 연산자를 오버로딩하면 개체에 둘러싸인 배열의 값을 간단하고 친숙한 방법으로 얻을 수 있으며 소스 코드의 가독성 향상에도 도움이 됩니다. 예를 들어, 지정된 위치에서 문자열의 기호에 대한 액세스를 제공해야 합니다. MQL5의 문자열은 별도의 string 유형으로, 심볼의 배열이 아니지만 오버로드된 인덱싱 작업을 통해 생성된 CString 클래스에서 간단하고 투명한 작업을 제공할 수 있습니다:

```
//+------------------------------------------------------------------+
//| 클래스의 심볼 배열에서와 같이 문자열의 심볼에 액세스합니다.              |
//+------------------------------------------------------------------+
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 문자열에서 심볼을 수신하기 위한 배열
   int     x[]={ 19,4,18,19,27,14,15,4,17,0,19,14,17,27,26,28,27,5,14,
                 17,27,2,11,0,18,18,27,29,30,19,17,8,13,6 };
   CString str("abcdefghijklmnopqrstuvwxyz[ ]CS");
   string  res;
//--- str 변수의 심볼을 사용하여 구문을 작성합니다
   for(int i=0,n=ArraySize(x);i<n;i++)
     {
      res+=ShortToString(str[x[i]]);
     }
//--- 결과 보기
   Print(res);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

인덱싱 작업의 오버로드의 또 다른 예는 매트릭스를 사용한 연산입니다. 행렬은 2차원 동적 배열을 나타내며 배열 크기는 미리 정의되지 않습니다. 따라서 두 번째 차원의 크기를 지정하지 않고 양식 array[][]의 배열을 선언한 다음 이 배열을 매개 변수로 전달할 수 없습니다. 가능한 솔루션은 CRow 클래스 개체의 배열을 포함하는 특수 클래스 CMatrix입니다.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 행렬의 덧셈 및 곱셈 연산
   CMatrix A(3),B(3),C();
//--- 행 배열 준비
   double a1[3]={1,2,3}, a2[3]={2,3,1}, a3[3]={3,1,2};
   double b1[3]={3,2,1}, b2[3]={1,3,2}, b3[3]={2,1,3};
//--- 행렬 채우기
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- Experts 로그에 행렬 출력
   Print("---- 행렬 A의 요소");
   Print(A.String());
   Print("---- 행렬 B의 요소");
   Print(B.String());

//--- 행렬의 덧셈
   Print("---- 행렬 A와 B의 합");
   C=A+B;
//--- 형식 지정된 문자열 표현 출력
   Print(C.String());

//--- 행렬의 곱셈
   Print("---- 행렬 A와 B의 곱");
   C=A*B;
   Print(C.String());

//--- 이제 동적 어레이 matrix[i][j] 스타일에서 값을 가져오는 방법을 보여 줍니다.
   Print("행렬 C의 값을 요소별로 출력");
//--- 행렬 행(CRow 객체)을 루프로 통과
   for(int i=0;i<3;i++)
     {
      string com="| ";
      //--- 행렬에서 값에 대한 행 서식
      for(int j=0;j<3;j++)
        {
         //--- 행과 열의 수로 행렬 요소 가져오기
         double element=C[i][j];// [i] - 배열 m_rows[]의 CRow 액세스 ,
                                // [j] - CRow에서 인덱싱 연산자가 오버로드되었습니다.
```

© 2000-2025, MetaQuotes Ltd.

```
         com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
      com+="|";
      //--- 행의 값을 출력
      Print(com);
    }
  }
//+------------------------------------------------------------------+
//| 클래스 "Row"                                                     |
//+------------------------------------------------------------------+
class CRow
  {
private:
   double            m_array[];
public:
   //--- 생성자 및 소멸자
                     CRow(void)           { ArrayResize(m_array,0);     }
                     CRow(const CRow &r)  { this=r;                     }
                     CRow(const double &array[]);
                    ~CRow(void){};
   //--- 행의 요소 수
   int               Size(void) const     { return(ArraySize(m_array));}
   //--- 값이 포함된 문자열을 반환합니다
   string            String(void) const;
   //--- 인덱싱 연산자
   double            operator[](int i) const  { return(m_array[i]);    }
   //--- 대입 연산자
   void              operator=(const double  &array[]); // 배열
   void              operator=(const CRow & r);         // 다른 CRow 객체
   double            operator*(const CRow &o);          // 곱셈을 위한 CRow 객체
  };
//+------------------------------------------------------------------+
//| 배열을 사용하여 행을 초기화하는 생성자                           |
//+------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
  {
   int size=ArraySize(array);
//--- 배열이 비어 있지 않은 경우
   if(size>0)
     {
      ArrayResize(m_array,size);
      //--- 값으로 채우기
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
    }
//---
  }
//+------------------------------------------------------------------+
//| 배열에 대한 할당 작업                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0) return;
//--- 배열을 값으로 채웁니다
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| CRow에 대한 할당 작업                                            |
//+------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0) return;
//--- 배열을 값으로 채웁니다
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------+
//| 다른 행의 곱셈 연산자                                            |
//+------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- 검증
   int size=Size();
   if(size!=o.Size() || size==0)
     {
      Print(__FUNCSIG__,": 두 행렬을 곱하지 못했습니다. 크기가 다릅니다.");
      return(res);
     }
//--- 요소별로 어레이를 곱하고 결과물 더하기
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
//--- 결과
   return(res);
  }
//+------------------------------------------------------------------+
//| 형식이 지정된 문자열 표현을 반환합니다                           |
//+------------------------------------------------------------------+
string CRow::String(void) const
  {
   string out="";
//--- 배열 크기가 0보다 큰 경우
   int size=ArraySize(m_array);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 0이 아닌 배열 요소만 사용합니다
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- 문자열에 대한 값 수집
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
    }
//--- 결과
   return(out);
  }
//+------------------------------------------------------------------+
//| Class "Matrix"                                                   |
//+------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- 생성자 및 소멸자
                     CMatrix(void);
                     CMatrix(int rows)  { ArrayResize(m_rows,rows);             }
                    ~CMatrix(void){};
   //--- 행렬 크기 가져오기
   int               Rows()       const { return(ArraySize(m_rows));            }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //--- 열의 값을 CRow 행 형식으로 반환합니다
   CRow              GetColumnAsRow(const int col_index) const;
   //--- 행렬 값이 있는 문자열을 반환합니다
   string            String(void) const;
   //--- 인덱싱 연산자가 번호별로 문자열을 반환합니다
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));        }
   //--- 덧셈 연산자
   CMatrix           operator+(const CMatrix &m);
   //--- 곱셈 연산자
   CMatrix           operator*(const CMatrix &m);
   //--- 대입 연산자
   CMatrix          *operator=(const CMatrix &m);
  };
//+------------------------------------------------------------------+
//| 기본 생성자, 크기가 0인 행 배열을 만듭니다                       |
//+------------------------------------------------------------------+
CMatrix::CMatrix(void)
  {
//--- 행렬의 행 수 0
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------+
//| 열 값을 CRow 형식으로 반환합니다                                 |
//+------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- 열에서 값을 가져오는 변수
   CRow row();
//--- 행렬의 행 수
   int rows=Rows();
//--- 행 수가 0보다 크면 작업을 실행합니다
   if(rows>0)
     {
      //--- 인덱스 col_index 가 있는 열의 값을 수신할 배열
      double array[];
      ArrayResize(array,rows);
      //--- 배열 채우기
      for(int i=0;i<rows;i++)
        {
         //--- i 행의 열 수 확인 - 배열의 경계를 초과할 수 있습니다
         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__,": Error! Column number ",col_index,"> row size ",i);
            break; // 행이 초기화되지 않은 객체가 됩니다
           }
         array[i]=this[i][col_index];
        }
      //--- 배열 값을 기반으로 CRow 행 생성
      row=array;
     }
//--- 결과
   return(row);
  }
//+------------------------------------------------------------------+
//| 두 행렬의 합                                                     |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- 전달된 행렬의 행 및 열 수
   int cols=m.Cols();
   int rows=m.Rows();
//--- 추가 결과를 수신할 행렬
   CMatrix res(rows);
//--- 행렬의 크기가 일치해야 합니다
   if(cols!=Cols() || rows!=Rows())
     {
      //--- 덧셈 불가
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCSIG__,": 두 행렬을 추가하지 못했습니다. 크기가 다릅니다");
      return(res);
     }
//--- 보조 배열
   double arr[];
   ArrayResize(arr,cols);
//--- 추가할 행으로 이동
   for(int i=0;i<rows;i++)
     {
      //--- 배열에 행렬 문자열을 추가한 결과 쓰기
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- 행렬 행에 배열 배치
      res[i]=arr;
     }
//--- 행렬 덧셈의 결과를 반환
   return(res);
  }
//+------------------------------------------------------------------+
//| 두 행렬의 곱                                                     |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- 첫 번째 행렬의 열 수, 행렬에 전달된 행 수
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- 추가 결과를 수신하는 매트릭스
   CMatrix res(rows1);
//--- 행렬은 조직되어야 합니다
   if(cols1!=rows2)
     {
      //--- 곱셈 불가
      Print(__FUNCSIG__,": 두 행렬을 곱하지 못했습니다. 형식이 호환되지 않습니다 "
            "- 첫 번째 요인의 열 수는 두 번째 요인의 행 수와 같아야 합니다");
      return(res);
     }
//--- 보조 배열
   double arr[];
   ArrayResize(arr,cols1);
//--- 곱셈 행렬의 행을 채웁니다
   for(int i=0;i<rows1;i++)// 행을 통과합니다
     {
      //--- 수신 배열 재설정
      ArrayInitialize(arr,0);
      //--- 행의 요소 통과
```

© 2000-2025, MetaQuotes Ltd.

```
      for(int k=0;k<cols1;k++)
        {
         //--- CRow의 에 행렬 m의 열 k 값을 취합니다
         CRow column=m.GetColumnAsRow(k);
         //--- 두 행을 곱하고 i 번째 요소에 벡터의 스칼라 곱하기 결과를 기록합니다
         arr[k]=this[i]*column;
        }
      //--- 행렬의 i 행에 배열 arr[] 배치
      res[i]=arr;
     }
//--- 두 행렬의 곱을 반환합니다
   return(res);
  }
//+------------------------------------------------------------------+
//| 할당 작업                                                        |
//+------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
//--- 행 수 찾기 및 설정
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- 전달된 행렬의 행 값으로 행을 채웁니다
   for(int i=0;i<rows;i++) this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------+
//| 행렬의 문자열 표현                                               |
//+------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//--- 문자열로 형성
   for(int i=0;i<rows;i++)
     {
      out=out+this[i].String()+"\r\n";
     }
//--- 결과
   return(out);
  }
```

더 보기

오버로드, 산술 연산, 오버로드 함수, 우선 순위 규칙

© 2000-2025, MetaQuotes Ltd.

외부 함수에 대한 설명

다른 모듈에 정의된 외부 기능은 명시적으로 설명해야 합니다. 설명에는 반환된 유형, 함수 이름 및 입력 매개 변수의 유형이 포함됩니다. 이러한 설명이 없으면 프로그램을 컴파일, 빌드 또는 실행할 때 오류가 발생할 수 있습니다. 외부 개체를 설명할 때 모듈을 나타내는 키워드 *#import*를 사용합니다.

예제:

```
#import "user32.dll"
  int     MessageBoxW(int hWnd ,string szText,string szCaption,int nType);
  int     SendMessageW(int hWnd,int Msg,int wParam,int lParam);
#import "lib.ex5"
  double  round(double value);
#import
```

가져오기 기능을 사용하면 외부 DLL 또는 컴파일된 EX5 라이브러리에서 호출되는 기능을 쉽게 설명할 수 있습니다. 라이브러리 속성을 가진 ex5 파일로 컴파일됩니다. EX5 라이브러리에서는 내보내기 한정자로 설명된 함수만 가져올 수 있습니다.

DLL 및 EX5 라이브러리는 함께 가져올 경우 해당 라이브러리가 위치한 디렉토리에 관계없이 서로 다른 이름을 가져야 합니다. 가져온 모든 함수는 라이브러리의 "파일 이름"에 해당하는 범위 지정을 가집니다.

예제:

```
#import "kernel32.dll"
   int GetLastError();
#import "lib.ex5"
   int GetLastError();
#import

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // class method 호출
      Print(::GetLastError());         // MQL5 함수 호출
      Print(kernel32::GetLastError()); // kernel32.dll 에서 DLL 라이브러리 함수 호출
      Print(lib::GetLastError());      // lib.exe5에서 EX5 라이브러리 함수 호출
     }
  };

void OnStart()
  {
   CFoo foo;
   foo.func();
  }
```

더 보기

© 2000-2025, MetaQuotes Ltd.

오버로드, 가상 함수, 다형성

© 2000-2025, MetaQuotes Ltd.

내보내기 함수

추출 사후 수정자를 사용하여 mql5 프로그램에 선언된 함수를 다른 mql5 프로그램에서 사용할 수 있습니다. 이러한 기능을 내보내기 가능이라고 하며 컴파일 후 다른 프로그램에서 호출할 수 있습니다.

```
int Function() export
  {
  }
```

이 한정자는 컴파일러에서 이 ex5 파일에서 내보낸 EX5 함수의 테이블에 함수를 추가하도록 명령합니다. 다른 mql5 프로그램에서는 이러한 한정자가 있는 함수만 액세스("가시적")할 수 있습니다.

라이브러리 속성은 컴파일러에게 EX5 파일이 라이브러리가 되고 컴파일러는 이를 EX5 헤더에 표시합니다.

추출할 수 있는 함수로 계획된 모든 함수는 *내보내기* 한정자로 표시되어야 합니다.

더 보기

오버로드, 가상 함수, 다형성

© 2000-2025, MetaQuotes Ltd.

이벤트 핸들링 함수

MQL5 언어는 일부 사전 정의된 이벤트의 처리를 제공합니다. 이러한 이벤트를 처리하기 위한 함수는 MQL5 프로그램에 정의되어야 합니다. 함수 이름, 반환 유형, 매개변수 구성(있는 경우) 및 해당 유형은 이벤트 핸들러 함수에 대한 설명과 엄격히 일치해야 합니다.

클라이언트 터미널의 이벤트 핸들러는 반환 값 유형 및 매개 변수 유형별로 이 이벤트 또는 해당 이벤트를 처리하는 함수를 식별합니다. 아래 설명에 해당하지 않는 다른 매개변수가 해당 함수에 지정되거나 다른 반환 유형이 지정된 경우 해당 함수는 이벤트 핸들러로 사용되지 않습니다.

OnStart

OnStart() 함수는 Start 이벤트 핸들러이며, 이는 자동으로 **오직 스크립트** 실행을 위해 작동합니다. 이는 매개 변수가 없는 void 유형이어야 합니다:

```
void OnStart();
```

OnStart() 함수의 경우, 반환 값으로 int 유형이 명시될 수 있습니다.

OnInit

OnInit() 함수는 Init 이벤트 핸들러입니다. 이는 매개 변수가 없는 void 또는 int 유형이어야 합니다:

```
void OnInit();
```

Init 이벤트는 Expert Advisor 또는 지표가 다운로드된 후 즉시 생성되며 스크립트에 대해서는 생성되지 않습니다. OnInit() 함수는 초기화에 사용됩니다. OnInit()의 반환 값에 int 타입이 있는 경우 0이 아닌 반환 코드는 실패한 초기화를 의미하며, Deinit 이벤트가 초기화 취소 사유 코드 REASON_INITFAILED와 함께 생성됩니다.

INIT_FAILED를 반환하면 EA가 차트에서 강제로 언로드됩니다.

INIT_FAILED를 반환하면 지표가 차트에서 언로드되지 않습니다. 차트에 남아 있는 지표는 작동하지 않습니다. —이벤트 핸들러는 지표에서 호출되지 않습니다.

Expert Advisor의 입력 매개 변수를 최적화하려면 ENUM_INIT_RETCODE 열거 값을 반환 코드로 사용하는 것이 좋습니다. 이 값은 가장 적절한 테스팅 에이전트 선택을 포함하여 최적화 과정을 구성하는 데 사용됩니다. 테스트를 시작하기 전에 Expert Advisor를 초기화하는 동안 TerminalInfoInteger() 함수를 사용하여 에이전트의 구성 및 리소스(코어 수, 사용 가능한 메모리 양 등)에 대한 정보를 요청할 수 있습니다. 수집된 정보에 따라 Expert Advisor를 최적화하는 동안 이 테스트 에이전트를 사용하도록 허용하거나 사용을 거부할 수 있습니다.

ENUM_INIT_RETCODE

식별자	설명
INIT_SUCCEEDED	Expert Advisor의 성공적인 초기화를 통해 테스트를 계속할 수 있습니다. 이 코드는 Null 값과 동일합니다 – Expert Advisor가 테스터에서 성공적으로 초기화되었습니다.
INIT_FAILED	초기화 실패; 치명적인 오류로 인해 테스트를 계속할 필요가 없습니다. 예를 들어 Expert Advisor 작업에 필요한 지표를 만들지 못

© 2000-2025, MetaQuotes Ltd.

식별자	설명
	했습니다. 이 반환 값이 0이 아닌 다른 값이라면 – 테스터에서 Expert Advisor를 초기화하지 못했다는 것을 뜻합니다.
INIT_PARAMETERS_INCORRECT	이 값은 입력 매개 변수의 잘못된 집합을 의미합니다. 이 반환 코드가 포함된 결과 문자열은 일반 최적화 테이블에서 빨간색으로 강조 표시됩니다. Expert Advisor의 지정된 매개 변수 집합에 대한 테스트가 실행되지 않으며 에이전트는 새 작업을 자유롭게 받을 수 있습니다. 이 값을 수신하면 전략 테스터가 이 작업을 다른 에이전트에 전달하여 재시도할 수 없습니다.
INIT_AGENT_NOT_SUITABLE	초기화 중 오류가 발생하지 않지만, 어떤 이유로 인해 에이전트가 테스트에 적합하지 않습니다. 예를 들어 메모리 부족, OpenCL 서포트 없음 등이 있습니다. 이 코드가 반환된 후 에이전트는 이 최적화가 끝날 때까지 작업을 수신하지 않습니다.

void 유형의 OnInit() 함수는 항상 성공적인 초기화를 나타냅니다.

OnDeinit

OnDeinit() 함수는 초기화 취소 중에 호출되며, 이는 Deinit 이벤트 핸들러입니다. 이 매개 변수는 void 유형으로 선언되어야 하며 const int 유형의 매개 변수가 하나 있어야 하며, 이는 초기화 해제 사유 코드를 포함해야 합니다. 만약 다른 유형이 선언된 경우, 컴파일러는 경고를 생성하지만 함수는 호출되지 않습니다. 스크립트의 경우 Deinit 이벤트가 생성되지 않으므로 OnDeinit() 함수를 스크립트에서 사용할 수 없습니다.

```
void OnDeinit(const int reason);
```

Deinit 이벤트는 다음과 같은 경우 Expert Advisor 및 지표에 대해 생성됩니다:

- mql5 프로그램이 부착된 심볼 또는 차트 기간의 변경으로 인한 재초기화 전;
- 입력 매개 변수 변경으로 인한 재초기화 전;
- mql5 프로그램을 언로드하기 전.

OnTick

NewTick 이벤트는 Expert Advisor 만 Expert Advisor가 부착된 차트에 심볼에 대한 새 틱이 수신될 때 생성됩니다. 커스텀 지표 또는 스크립트에 OnTick() 함수를 정의하는 것은 NewTick 이벤트가 생성되지 않기 때문에 무용지물입니다.

Tick 이벤트는 Expert Advisor에 대해서만 생성되지만 Expert Advisor에 대해 NewTick 이벤트가 생성될 뿐만 아니라 Timer, BookEvent 및 ChartEvent 이벤트 이벤트도 생성되므로 Expert Advisor에 OnTick() 함수가 필요하지는 않습니다. 이것은 매개 변수가 없는 void 유형으로 선언되어야 합니다:

```
void OnTick();
```

OnTimer

© 2000-2025, MetaQuotes Ltd.

OnTimer() 함수는 Expert Advisor 및 지표에 대해서만 시스템 타이머에 의해 생성되는 타이머 이벤트가 발생할 때 호출되며 -이는 스크립트에서 사용할 수 없습니다. 이벤트 발생 빈도는 EventSetTimer() 함수가 수신할 이 이벤트에 대한 알림을 구독할 때 설정됩니다.

EventKillTimer() 함수를 사용하여 특정 Expert Advisor에 대한 타이머 이벤트 수신을 취소할 수 있습니다. 함수는 매개 변수 없이 void 유형으로 정의되어야 합니다:

```
void OnTimer();
```

OnInit() 함수에서 EventSetTimer() 함수를 한 번 호출하고 OnDeinit() 함수에서 EventKillTimer() 함수를 한 번 호출하는 것이 좋습니다.

모든 Expert Advisor는 물론 모든 지표가 자체 타이머로 작동하며 이 타이머로부터만 이벤트를 수신합니다. mql5 프로그램 작동이 중지되는 즉시 타이머가 생성되었지만 EventKillTimer() 함수에 의해 비활성화되지 않은 경우 강제로 삭제됩니다.

OnTrade

이 함수는 거래 이벤트가 발생할 때 호출되는데, 이는 주문 및 포지션 오픈, 주문 내역 및 거래 내역을 변경시 나타납니다. 거래 활동(보류 주문 오프닝, 포지션 오픈/클로징, 설정 중지, 보류 주문 트리거 등)이 수행되면 주문 및/또는 거래의 내역과 현재 주문 목록이 그에 따라 변경됩니다.

```
void OnTrade();
```

유저는 (거래전략조건에 따라 요구되는 경우) 그러한 이벤트가 접수되었을 때 거래계좌 상태에 대한 검증을 코드에 독립적으로 구현해야 합니다. OrderSend() 함수 호출이 성공적으로 완료되어 true 값을 반환한 경우, 거래 서버가 실행을 위해 주문을 대기열에 넣고 티켓 번호를 할당했음을 의미합니다. 서버가 이 주문을 처리하는 즉시 거래 이벤트가 생성됩니다. 또한 사용자가 티켓 값을 기억하면 OnTrade() 이벤트 처리 중에 이 값을 사용하여 주문에 무슨 일이 발생했는지 확인할 수 있습니다.

OnTradeTransaction

거래 계정에 대해 몇 가지 명확한 작업을 수행하면 상태가 변경됩니다. 그러한 액션은 다음을 포함합니다:

- OrderSend 및 OrderSendAsync 함수를 사용하여 클라이언트 터미널의 모든 MQL5 응용 프로그램에서 거래 요청을 전송하고 해당 추가 실행을 수행;
- 터미널 그래픽 인터페이스 및 추가 실행을 통해 거래 요청 전송;
- 서버에서 주문 보류 및 주문 활성화 중지;
- 거래 서버 측에서 작업 수행.

이러한 조치의 결과로 다음과 같은 매매 거래가 수행됩니다:

- 거래 요청 처리;
- 오픈 주문 변경;
- 주문 내역 변경;
- 거래 내역 변경;
- 포지션 변경.

예를 들어, 시장 매수 주문을 보낼 때 처리되고, 계정에 대한 적절한 매수 주문이 생성되며, 주문이 실행되어 열린 주문 목록에서 삭제되고, 주문 기록에 추가되고, 적절한 거래가 기록에 추가되고, 새로운 포지

© 2000-2025, MetaQuotes Ltd.

션이 생성됩니다. 이 모든 행동은 거래 트랜잭션입니다. 그러한 거래가 터미널에 도착하는 것은 TradeTransaction 이벤트 입니다. 이는 OnTradeTransaction 핸들러를 호출합니다

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // 거래 트랜잭션 구조
   const MqlTradeRequest&        request,      // 요청 구조
   const MqlTradeResult&         result        // 결과 구조
   );
```

핸들러에는 세 가지 매개 변수가 있습니다:

- **trans** – 이 매개 변수는 MqlTradeTransaction 구조를 가져오는데, 이는 거래 계정에 적용되는 거래 트랜잭션을 설명합니다;
- **request** – 이 매개 변수는 거래 요청을 설명하는 MqlTradeRequest 구조를 가져옵니다;
- **result** – 이 매개 변수는 거래 요청 실행 결과를 설명하는 MqlTradeResult 구조를 가져옵니다.

마지막 두 **요청** 및 **결과** 매개 변수는 TRADE_TRANSACTION_REQUEST 유형 트랜잭션에 대한 값으로 채워지며 *type* 매개 변수(**trans** 변수)에서 트랜잭션 데이터를 수신할 수 있습니다. 이 경우, *request_id* 필드(**result** 변수의) **request** trade request ID를 포함하며, 이 ID의 실행 후에는 거래 트랜잭션 (**trans** 변수에 설명된)이 수행됩니다. 요청 ID를 사용하면 수행된 작업(OrderSend 또는 OrderSendAsync 함수 호출)을 OnTradeTransaction()로 보낸 작업의 결과와 연결할 수 있습니다.

터미널에서 수동으로 전송되거나 OrderSend()/OrderSendAsync() 함수를 통해 전송된 거래 요청 한 번으로 거래 서버에서 여러 개의 연속 트랜잭션이 발생할 수 있습니다. 이러한 거래의 터미널 도착 우선순위는 보장되지 않습니다. 따라서 거래 알고리즘을 개발할 때 트랜잭션 그룹이 하나씩 차례로 도착할 것이라고 예상해서는 안 됩니다.

- 모든 유형의 거래 트랜잭션은 ENUM_TRADE_TRANSACTION_TYPE 열거값에 설명되어 있습니다.
- 거래 트랜잭션을 설명하는 MqlTradeTransaction 구조는 거래 유형에 따라 다양한 방식으로 채워집니다. 예를 들어 TRADE_TRANSACTION_REQUEST 유형 트랜잭션의 경우 유형 필드(트레이드 트랜잭션 타입)만 분석해야 합니다. 추가 데이터에 대해 OnTradeTransaction 함수의 두 번째 및 세 번째 매개 변수(요청 및 결과)를 분석해야 합니다. 자세한 내용은 "거래 트랜잭션 구조"를 참조하십시오.
- 거래 설명은 주문, 거래 및 포지션(예: 의견)과 관련하여 사용 가능한 모든 정보를 전달하지는 않습니다. OrderGet*, HistoryOrderGet*, HistoryDealGet* 및 PositionGet* 함수는 더 확장된 정보를 얻기 위해 사용해야 합니다.

클라이언트 계정에 거래 트랜잭션을 적용한 후에는 터미널 거래 대기열에 일관되게 배치되며, 터미널에 도착하는 순서대로 OnTradeTransaction 진입 포인트로 일관되게 전송됩니다.

OnTradeTransaction 핸들러를 사용하여 Expert Advisor가 거래 트랜잭션을 처리할 때 터미널은 새로 도착한 거래 트랜잭션을 계속 처리합니다. 따라서 OnTradeTransaction 작업 중에 거래 계정의 상태가 이미 변경될 수 있습니다. 예를 들어, MQL5 프로그램이 새 순서를 추가하는 이벤트를 처리하는 동안 이 프로그램이 실행되어 열린 순서 목록에서 삭제되고 기록으로 이동할 수 있습니다. 또한 어플리케이션에는 이러한 이벤트가 통지됩니다.

트랜잭션 대기열 길이는 1024개의 요소로 구성됩니다. OnTradeTransaction이 새 트랜잭션을 너무 오래 처리하면 대기열에 있는 이전 트랜잭션이 새 트랜잭션으로 대체될 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

- 일반적으로 OnTrade 및 OnTradeTransaction 요청 수의 정확한 비율은 없습니다. 하나의 OnTrade 호출은 하나 이상의 OnTradeTransaction 호출에 해당합니다.
- OnTrade는 적절한 OnTradeTransaction 호출 후에 호출됩니다.

OnTester

OnTester() 함수는 선택한 간격에 대한 Expert Advisor의 기록 테스트가 끝난 후 자동으로 생성되는 Tester 이벤트의 핸들러입니다. 함수는 매개 변수 없이 double 유형으로 정의되어야 합니다:

```
double OnTester();
```

이 함수는 OnDeinit() 호출 직전에 호출되며 반환 값의 유형이 같습니다 – double. OnTester()는 Expert Advisors의 테스트 중에만 사용 가능합니다. 주요 목적은 입력 파라미터의 유전적 최적화에서 Custom max 기준으로 사용되는 특정 값을 계산하는 것입니다.

유전적 최적화에서는 내림차순 정렬이 한 세대 내의 결과에 적용됩니다. 즉, 최적화 기준의 관점에서 최상의 결과는 가장 큰 값을 가지는 것입니다(OnTester 함수에 의해 반환되는 커스텀 최대 최적화 기준 값이 고려됨). 이러한 정렬에서 최악의 값은 끝에 배치되고 더 멀리 던져지며 다음 세대의 형성에는 참여하지 않는 것입니다.

OnTesterInit

전략 테스터에서 최적화하기 전에 필요한 작업을 수행하기 위해 TesterInit 이벤트가 발생하면 EA에서 이 기능을 호출합니다. 두 가지 함수 유형이 있습니다.

결과를 반환하는 **버전**

```
int  OnTesterInit(void);
```

반환 값

int 타입 값, 0은 최적화가 시작되기 전에 차트에서 EA가 성공적으로 초기화되었음을 의미합니다.

실행 결과를 반환하는 OnTesterInit() 호출은 프로그램 초기화를 허용할 뿐만 아니라 초기 최적화 중지 시 오류 코드를 반환하므로 사용을 권장합니다. INIT_SUCCEEDED (0) 이외의 값이 반환되면 오류가 발생하며 최적화가 시작되지 않습니다.

결과 반환 없는 버전은 이전 코드와의 호환성을 위해서만 유지됩니다. 이용을 권장하지 않습니다

```
void  OnTesterInit(void);
```

최적화가 시작되면 OnTesterDeinit() 또는 OnTesterPass() 핸들러가 있는 Expert Advisor가 테스터에 지정된 기호와 기간이 포함된 별도의 터미널 차트에 자동으로 로드되고 TesterInit 이벤트를 수신합니다. 이 함수는 최적화 결과 처리를 위한 최적화 시작 전 Expert Advisor 초기화에 사용됩니다.

OnTesterPass

OnTesterPass() 함수는 전략 테스터에서 Expert Advisor 최적화 중에 프레임이 수신되면 자동으로 생성되는 TesterPass 이벤트의 핸들러입니다. 이 함수는 반드시 void 타입으로 정의되어야 합니다. 이것에는 매개 변수가 없습니다:

```
void OnTesterPass();
```

© 2000-2025, MetaQuotes Ltd.

OnTesterPass() 핸들러가 있는 Expert Advisor는 테스트를 위해 지정된 심볼/주기가 있는 별도의 터미널 차트에 자동으로 로드되며 최적화 중에 프레임이 수신되면 TesterPass 이벤트를 가져옵니다. 이 함수는 완료를 기다리지 않고 "즉석에서" 최적화 결과를 동적으로 처리하는 데 사용됩니다. 프레임은 FrameAdd() 함수를 사용하여 추가되며, 이 함수는 OnTester() 핸들의 단일 패스가 끝난 후 호출할 수 있습니다.

OnTesterDeinit

OnTesterDeinit()는 전략 테스터에서 Expert Advisor 최적화가 종료된 후 자동으로 생성되는 TesterDeinit 이벤트의 핸들러입니다. 이 함수는 반드시 void 타입으로 정의되어야 합니다. 이것에는 매개 변수가 없습니다:

```
void OnTesterDeinit();
```

TesterDeinit() 핸들러가 있는 Expert Advisor는 최적화를 시작할 때 차트에 자동으로 로드되고 완료 후 TesterDeinit을 수신합니다. 이 함수는 모든 최적화 결과를 최종 처리하는 데 사용됩니다.

OnBookEvent

OnBookEvent() 함수는 BookEvent 핸들러입니다. BookEvent는 시장 심도가 변경될 때 Expert Advisor 및 지표에 대해 생성됩니다. 이는 void 형식이어야 하며 문자열 유형의 매개 변수가 하나 있어야 합니다:

```
void OnBookEvent (const string& symbol);
```

심볼에 대한 BookEvent 이벤트를 수신하려면 MarketBookAdd() 함수를 사용하여 이 심볼에 대한 이벤트를 수신하려면 사전 구독만 하면 됩니다. 특정 심볼에 대한 BookEvent 이벤트 수신을 취소하려면, MarketBookRelease()를 호출하십시오.

다른 이벤트와는 달리, BookEvent 이벤트는 방송됩니다. 즉, 하나의 Expert Advisor 구독자가 MarketBookAdd를 사용하여 BookEvent 이벤트 수신하면 OnBookEvent() 핸들러가 있는 다른 모든 Expert Advisor가 이 이벤트를 수신합니다. 따라서 *const string& symbol* 매개 변수로 핸들러에 전달되는 심볼명을 분석해야 합니다.

OnChartEvent

OnChartEvent()는 ChartEvent 이벤트 그룹의 핸들러입니다:

- CHARTEVENT_KEYDOWN — 차트 창에 포커스가 맞춰진 경우 키 입력 이벤트;
- CHARTEVENT_MOUSE_MOVE — 마우스 이동 이벤트 및 마우스 클릭 이벤트 (CHART_EVENT_MOUSE_MOVE=true 가 차트에 대해 설정된 경우);
- CHARTEVENT_OBJECT_CREATE — 그래픽 객체 생성 이벤트 (CHART_EVENT_OBJECT_CREATE=true 가 차트에 대해 설정된 경우);
- CHARTEVENT_OBJECT_CHANGE — 특성 다이얼로그를 통해 객체 특성이 변경될 경우;
- CHARTEVENT_OBJECT_DELETE — 그래픽 객체 삭제 이벤트 (CHART_EVENT_OBJECT_DELETE=true 가 차트에 대해 설정된 경우);
- CHARTEVENT_CLICK — 차트에서 마우스를 클릭하는 이벤트;
- CHARTEVENT_OBJECT_CLICK — 차트에 속한 그래픽 객체를 마우스로 클릭하는 이벤트;
- CHARTEVENT_OBJECT_DRAG — 마우스를 사용한 그래픽 객체 이동 이벤트;
- CHARTEVENT_OBJECT_ENDEDIT — 그 LabelEdit 그래픽 개체의 입력 상자에서 끝난 텍스트 편집의 이벤트;

© 2000-2025, MetaQuotes Ltd.

- CHARTEVENT_CHART_CHANGE — 차트 변경 이벤트;
- CHARTEVENT_CUSTOM+n — 사용자 이벤트의 ID로, 여기서 n은 0에서 65535 사이의 범위입니다.
- CHARTEVENT_CUSTOM_LAST — 커스텀 이벤트의 마지막 허용 ID(CHARTEVENT_CUSTOM +65535).

이 함수는 Expert Advisor 및 지표에서만 호출할 수 있습니다. 함수는 4개의 매개 변수를 가진 보이드 유형이어야 합니다:

```
void OnChartEvent(const int id,          // Event ID
                  const long& lparam,    // long 타입 이벤트의 매개 변수
                  const double& dparam, // double 타입 이벤트의 매개 변수
                  const string& sparam  // string 타입 이벤트의 매개 변수
  );
```

각 이벤트 유형에 대해 OnChartEvent() 함수의 입력 매개 변수에는 이 이벤트 처리에 필요한 한정된 값이 있습니다. 이러한 매개 변수를 통과하는 이벤트 및 값은 아래 표에 나열되어 있습니다.

Event	id 매개 변수 값	lparam 매개변수 값	dparam 매개변수 값	sparam 매개변수 값
키 입력 이벤트	CHARTEVENT_KEYDOWN	누른 키의 코드	반복 횟수(사용자가 키를 누른 결과로 키 입력이 반복되는 횟수)	키보드 버튼의 상태를 설명하는 비트 마스크의 문자열 값
마우스 이벤트 (property CHART_EVENT_MOUSE_MOVE=true 가 차트에 대해 설정된 경우)	CHARTEVENT_MOUSE_MOVE	X 좌표	Y 좌표	마우스 버튼의 상태를 설명하는 비트 마스크의 문자열 값
그래픽 객체 생성 이벤트 (CHART_EVENT_OBJECT_CREATE =true 가 차트에 대해 설정된 경우)	CHARTEVENT_OBJECT_CREATE	—	—	생성된 그래픽 개체의 이름
속성 대화상자를 통한 객체 속성 변경 이벤트	CHARTEVENT_OBJECT_CHANGE	—	—	수정된 그래픽 개체의 이름
그래픽 객체 삭제 이벤트 (CHART_EVENT_OBJECT_DELETE =true 가 차트에 대해 설정된 경우)	CHARTEVENT_OBJECT_DELETE	—	—	삭제된 그래픽 개체의 이름
차트를 마우스로 클릭하는 이벤트	CHARTEVENT_CLICK	X 좌표	Y 좌표	—

© 2000-2025, MetaQuotes Ltd.

Event	id 매개 변수 값	lparam 매개변수 값	dparam 매개변수 값	sparam 매개변수 값
차트에 속한 그래픽 개체를 마우스로 클릭하는 이벤트	CHARTEVENT_OBJECT_CLICK	X 좌표	Y 좌표	이벤트가 발생한 그래픽 개체의 이름
마우스를 사용하여 그래픽 개체를 드래그하는 이벤트	CHARTEVENT_OBJECT_DRAG	—	—	이동된 그래픽 개체의 이름
레이블 편집 그래픽 객체의 입력 상자에서 완료된 텍스트 편집 이벤트	CHARTEVENT_OBJECT_ENDEDIT	—	—	다음의 이름: 텍스트 편집이 완료된 LabelEdit 그래픽 객체
차트 변경 이벤트	CHARTEVENT_CHART_CHANGE	—	—	—
N 넘버 이하의 유저 이벤트 ID	CHARTEVENT_CUSTOM+N	EventChartCustom() 함수에 의해 설정된 값	EventChartCustom() 함수에 의해 설정된 값	EventChartCustom() 함수에 의해 설정된 값

OnCalculate

OnCalculate() 함수는 Calculate 이벤트로 지표 값을 계산해야 하는 경우에만 커스텀 지표에서 호출됩니다. 이 문제는 일반적으로 지표가 계산된 기호에 대해 새 눈금이 수신될 때 발생합니다. 이 지표는 이 심볼의 가격 차트에 첨부할 필요가 없습니다.

OnCalculate() 함수는 int 타입으로만 반환합니다. 두 가지 가능한 정의가 있습니다. 한 지표 내에서 두 가지 버전의 함수를 모두 사용할 수 없습니다.

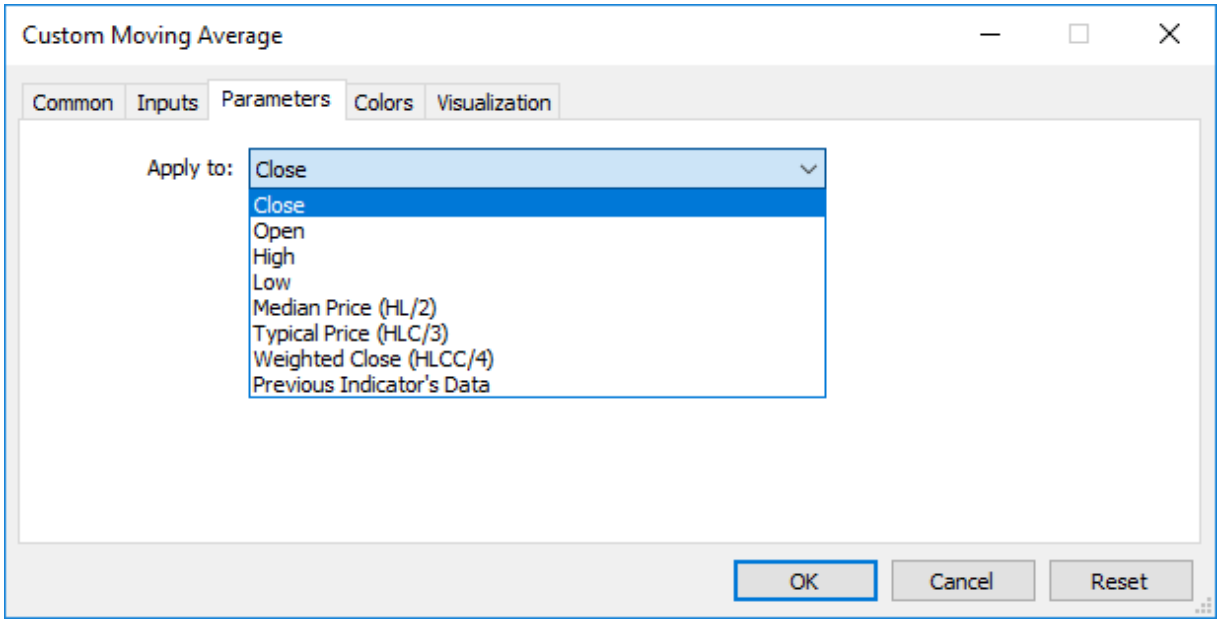
첫 번째 양식은 단일 데이터 버퍼에서 계산할 수 있는 지표를 위한 것입니다. 이러한 지표의 예로는 커스텀 이동 평균이 있습니다.

```
int OnCalculate (const int rates_total,      // price[] 배열의 크기
                 const int prev_calculated,  // 이전 호출에서 처리된 막대
                 const int begin,            // 중요한 데이터가 시작되는 곳
                 const double& price[]       // 계산할 배열
   );
```

price[] 배열로 시계열 중 하나 또는 일부 지표의 계산된 버퍼를 통과할 수 있습니다. price[] 배열에서 인덱싱의 방향을 결정하려면 ArrayGetAsSeries()를 호출하십시오. 기본값에 의존하지 않으려면 함께 작동해야 하는 배열에 대해 무조건 ArraySetAsSeries() 함수를 호출해야 합니다.

사용자가 지표를 시작할 때 "파라미터" 탭에서 price[] 배열로 사용할 시계열 또는 지표를 선택할 수 있습니다. 이렇게 하려면 "적용 대상" 필드의 드롭다운 목록에서 필요한 항목을 지정해야 합니다.

© 2000-2025, MetaQuotes Ltd.



다른 mql5 프로그램으로부터 커스텀 지표의 값을 수신하려면 후속 작업에 대한 지표 핸들을 반환하는 iCustom() 함수를 사용합니다. 적절한 price[] 배열 또는 다른 지표의 핸들을 지정할 수도 있습니다. 이 매개 변수는 커스텀 지표의 입력 변수 목록에서 마지막으로 전송되어야 합니다.
예:

```
void OnStart()
  {
//---
   string terminal_path=TerminalInfoString(STATUS_TERMINAL_PATH);
   int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving Average",13,0,
   if(handle_customMA>0)
      Print("handle_customMA = ",handle_customMA);
   else
      Print("열 수 없거나 EX5 파일이 아닙니다 '"+terminal_path+"\\MQL5\\Indicators\\"+"Cu
  }
```

이 예제에서, 전달돈 마지막 매개 변수는 (ENUM_APPLIED_PRICE 열거값 에서) PRICE_TYPICAL 값이며, 이는(High+Low+Close)/3 으로 얻은 일반적인 가격을 기반으로 커스텀 지표가 작성됨을 나타냅니다. 이 매개변수가 지정되지 않은 경우 지표는 PRICE_CLOSE 값(즉, 각 막대의 종가)을 기반으로 작성됩니다.

지표 핸들러가 price[] 배열을 지정하는 마지막 매개 변수로 전달되는 것을 보여주는 또 다른 예는 iCustom() 함수에 대한 설명에 나와 있습니다.

두 번째 형태는 계산에 두 개 이상의 시계열이 사용되는 다른 모든 지표에 사용됩니다.

```
int OnCalculate (const int rates_total,      // 입력 시계열의 크기
                 const int prev_calculated,  // 이전 호출에서 처리한 막대
                 const datetime& time[],     // 시간
                 const double& open[],       // 시가
                 const double& high[],       // 고가
                 const double& low[],        // 저가
                 const double& close[],      // 종가
```

© 2000-2025, MetaQuotes Ltd.

```
                const long& tick_volume[],  // 틱 볼륨
                const long& volume[],       // 실제 거래량
                const int& spread[]         // 스프레드
   );
```

open[], high[], low[] 및 close[]의 파라미터에는 현재 기간의 시가, 고가, 저가, 종가의 배열이 포함됩니다. time[] 매개 변수에는 시가가 있는 배열이 포함되며, spread[] 매개 변수에는 스프레드의 내역이 포함된 배열이 있습니다(거래된 증권에 대해 제공된 경우). volume[]과 tick_volume[]의 매개 변수에는 각각 거래 내역과 틱 볼륨 내역이 포함됩니다.

time[], open[], high[], low[], close[], tick_volume[], volume[] 및 spread[]의 인덱싱을 확인하려면, ArrayGetAsSeries()를 호출하십시오. 기본값에 의존하지 않으려면 해당 어레이에 대해 무조건적으로 ArraySetAsSeries() 함수를 호출해야 하며, 이는 함께 사용할 수 없습니다.

첫 번째 rates_total 매개 변수는 계산에 사용할 수 있는 막대의 수를 포함하며 차트에서 사용할 수 있는 막대의 수에 해당합니다.

OnCalculate()의 반환 값과 prev_calculated의 두 번째 입력 매개 변수 사이의 연결성에 유의해야 합니다. 함수 호출 중, prev_calculated 매개 변수는 OnCalculate()에 의해**반환**된 값을 포함하며, 이는 **이전** 호출 중의 것입니다. 따라서 이 함수의 이전 실행 이후 변경되지 않은 막대에 대한 반복적인 계산을 피하기 위해 커스텀 지표를 계산하는 경제적 알고리즘이 가능합니다.

이를 위해 일반적으로 현재 함수 호출의 막대 수를 포함하는 rates_total 매개변수의 값을 반환하는 것으로 충분합니다. OnCalculate() 가격 데이터의 마지막 호출이 변경된 경우(더 깊은 히스토리 다운로드 또는 히스토리 공백이 채워짐) 터미널에 의해 입력 매개변수 prev_calculated 값이 0으로 설정됩니다.

참고: OnCalculate가 0을 반환하면 클라이언트 터미널의 데이터 창에 지표 값이 표시되지 않습니다.

이것을 더 잘 이해하기 위해 아래에 첨부된 코드를 시작하는 것이 유용할 것입니다.

지표 예제:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Line
#property indicator_label1  "Line"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDarkBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 지표 버퍼
double         LineBuffer[];
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 매핑
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime& time[],
                const double& open[],
                const double& high[],
                const double& low[],
                const double& close[],
                const long& tick_volume[],
                const long& volume[],
                const int& spread[])
  {
//--- 현재 심볼 및 차트 기간에 사용할 수 있는 막대 수를 가져옵니다.
   int bars=Bars(Symbol(),0);
   Print("Bars = ",bars,", rates_total = ",rates_total,",  prev_calculated = ",prev_ca
   Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

더 보기

프로그램 실행, 클라이언트 터미널 이벤트, 이벤트 작업

© 2000-2025, MetaQuotes Ltd.

변수

변수 선언

변수를 사용하기 전에 선언해야 합니다. 변수를 식별하는 데 고유 이름이 사용됩니다. 변수를 선언하려면 변수 유형과 고유한 이름을 지정해야 합니다. 변수 선언은 연산자가 아닙니다.

단순 유형은 다음과 같습니다:

- char, short, int, long, uchar, ushort, uint, ulong – 정수;
- color – RGB 색상을 표현하는 정수;
- datetime – 날짜 및 시간, 1970년 1월 1일 0시 이후 초 수를 포함하는 부호 없는 정수;
- bool – 부울 값 *참*과 *거짓*;
- double – 배정밀도 부동 소수점 수;
- float – 단정도 부동 소수점 수;
- string – 문자열.

예제:

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

컴플렉스 또는 복합 유형:

구조는 다른 유형을 사용하여 구성된 복합 데이터 유형입니다.

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // 이전에 선언된 구조 MyTime의 변수
```

구조를 선언하기 전에는 구조 유형의 변수를 선언할 수 없습니다.

배열

배열은 동일한 유형 데이터의 인덱싱된 순서입니다:

```
int    a[50];        // 50개의 정수로 구성된 1차원 배열.
double m[7][50];     // 7개의 배열로 구성된 2차원 배열,
                     // 각각 50개의 숫자로 구성되어 있음.
MyTime t[100];       // MyTime과 같은 요소가 포함된 배열
```

© 2000-2025, MetaQuotes Ltd.

배열 색인은 정수만 사용할 수 있습니다. 4차원 배열만 허용됩니다. 배열 요소의 번호 지정은 0으로 시작합니다. 1차원 배열의 마지막 요소는 배열 크기보다 1 작은 숫자를 가집니다. 즉, 50개의 정수로 구성된 배열의 마지막 요소에 대한 호출이 a[49]로 나타납니다. 다차원 배열에 대해서도 마찬가지입니다: 차원은 0에서 치수 크기-1까지 인덱싱됩니다. 예제에서 2차원 배열의 마지막 요소는 m[6][49]로 나타납니다.

정적 배열을 시계열로 나타낼 수 없습니다. 즉, 배열 요소에 대한 액세스를 처음부터 설정하는 ArraySetAsSeries() 함수를 정적 배열에 적용할 수 없습니다. 시계열에서와 동일한 배열에 대한 액세스를 제공하려면 동적 배열 객체를 사용하십시오.

어레이 범위를 벗어난 액세스 시도가 있을 경우 실행 중인 하위 시스템에서 심각한 오류가 발생하고 프로그램이 중지됩니다.

배열 작업을 위한 내장 메서드

Array Functions 섹션의 함수와 내장 메서드를 사용하여 배열을 처리할 수 있습니다:

메서드	Analog	설명
void array.Fill(const scalar value, const int start_pos=0, const int count=-1);	ArrayFill, ArrayInitialize	지정된 값으로 배열을 채웁니다.
void array.Free();	ArrayFree	동적 배열 버퍼를 해제하고 0차원 크기를 0(zero)으로 설정합니다.
int array.Resize(const int range0_size, const int reserve); int array.Resize(const int range_sizes[], const int reserve);	ArrayResize	일차원 배열에서 새로운 크기를 설정합니다.
int array.Print();	ArrayPrint	저널에 단순형 배열 값을 표시합니다.
int array.Size(const int range=-1);	ArraySize, ArrayRange	전체 배열(범위=-1) 또는 지정된 배열 차원의 요소 수를 반환합니다.
bool array.IsDynamic();	ArrayIsDynamic	배열이 동적인지 체크합니다
bool array.IsIndicatorBuffer();		배열이 지표의 버퍼인지 체크합니다
bool array.IsSeries();	ArrayIsSeries	배열이 타임시리즈인지 체크합니다
bool array.AsSeries();	ArrayGetAsSeries	배열 인덱싱 방향을 체크합니다
bool array.AsSeries(const bool as_series);	ArraySetAsSeries	배열에서 인덱싱 방향을 설정합니다
int array.Copy(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCopy	배열 값을 다른 배열에 복사합니다
int array.Compare(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCompare	두 개의 단순 유형 배열 또는 사용자 정의 구조를 비교한 결과를 리턴합니다.
int array.Insert(const src_array[], const int dst_start, const int	ArrayInsert	소스 배열의 지정된 수의 요소를 지정된 인덱스에서 시작하여 수신하는 배

© 2000-2025, MetaQuotes Ltd.

메서드	Analog	설명
src_start, const int cnt);		열에 삽입합니다.
int array.Remove(const int start_pos, const int count);	ArrayRemove	지정된 인덱스로 시작하는 배열에서 지정된 수의 요소를 제거합니다.
int array.Reverse(const int start_pos, const int count);	ArrayReverse	지정된 인덱스로 시작하는 배열의 지정된 수의 요소를 반전시킵니다.
bool array.Swap(array& arr[]);	ArraySwap	콘텐츠를 동일한 다른 동적 배열과 교환합니다 type
void array.Sort(sort_function);	ArraySort	일차원을 기준으로 숫자형 배열을 정렬합니다.
int array.Search(scalar value, search_function);	ArrayBsearch	일차원 배열에서 먼저 감지된 첫 번째 요소의 인덱스를 반환합니다.
int array.Find((scalar value, search_function);		전달된 함수를 사용하여 배열에서 검색을 수행하고 최초로 감지된 요소의 인덱스를 반환합니다.
array array.Select(scalar value, search_function);		전달된 함수를 사용하여 배열에서 검색을 수행하고 감지된 모든 요소가 포함된 배열을 반환합니다.

액세스 지정자

액세스 지정자는 컴파일러에서 변수, 구조 또는 클래스의 멤버에 액세스하는 방법을 정의합니다.

const 지정자는 변수를 상수로 선언하며 런타임 중에 이 변수를 변경할 수 없습니다. 변수를 선언할 때 변수를 단일 초기화할 수 있습니다.

예제:

```
int OnCalculate (const int rates_total,      // price[] 배열의 크기
                 const int prev_calculated,  // 이전 호출에서 처리된 막대
                 const int begin,            // 중요한 데이터가 시작되는 곳
                 const double& price[]       // 계산할 배열
   );
```

구조물과 클래스의 멤버에 액세스하려면 다음 한정자를 사용합니다:

- public – 변수 또는 클래스 메서드에 제한 없이 액세스할 수 있습니다
- protected – 에서는 이 클래스의 메서드와 공개적으로 상속된 클래스의 메서드에서 액세스할 수 있습니다. 다른 접근은 불가합니다;
- private – 은 같은 클래스의 메서드에서만 변수 및 클래스 메서드에 액세스할 수 있도록 허용합니다.
- virtual – 는 클래스 메서드에만 적용되며(구조 메서드에는 적용되지 않음) 이 메서드는 클래스의 가상 함수 테이블에 배치되어야 함을 컴파일러에 알려줍니다.

저장 클래스

© 2000-2025, MetaQuotes Ltd.

이것들은 세가지 스토리지 클래스입니다: static, input 및 extern. 스토리지 클래스의 이러한 수정자는 해당 변수가 사전 할당된 메모리 영역에 분산되어 있음을 컴파일러에 명시적으로 나타내며, 이를 글로벌 풀이라고 합니다. 게다가, 이 수식어들은 변수 데이터의 특별한 처리를 나타냅니다. 로컬 수준에서 선언된 변수가 정적 변수가 아닌 경우 해당 변수의 메모리는 프로그램 스택에서 자동으로 할당됩니다. 정적 어레이가 선언된 블록의 가시성 영역을 벗어나면 정적 어레이에 할당된 메모리 여유도 자동으로 수행됩니다.

더 보기

데이터 유형, 유형의 캡슐화 및 확장성,변수 초기화, 가시성 범위 및 변수 수명, 객체 생성 및 삭제, 클래스의 정적 멤버

© 2000-2025, MetaQuotes Ltd.

로컬 변수

함수 안에서 선언된 변수는 로컬입니다. 로컬 변수의 범위는 선언된 함수 범위로 제한됩니다. 로컬 변수는 초기화 될 수 있습니다, 모든 표현식의 결과로써. 함수의 모든 호출은 로컬 변수를 초기화합니다. 로컬 변수는 해당 함수의 메모리 영역에 저장됩니다.

예제:

```
int somefunc()
  {
   int ret_code=0;
   ...
   return(ret_code);
  }
```

범위-변수의-는 변수를 참조할 수 있는 프로그램의 부분입니다. 내부 수준에서 블록 내부에 선언된 변수의 범위는 블록입니다. 블록 범위는 변수 선언으로 시작하여 마지막 오른쪽 꺽쇠로 끝납니다.

함수의 시작 부분에 선언된 로컬 변수에는 블록의 범위뿐만 아니라 로컬 변수인 함수 매개 변수도 있습니다. 모든 블록에는 변수 선언이 포함될 수 있습니다. 블록이 중첩되어 있고 외부 블록의 식별자가 내부 블록의 식별자와 이름이 같으면 내부 블록 작업이 끝날 때까지 외부 블록 식별자가 숨겨집니다.

예제:

```
void OnStart()
  {
//---
   int i=5;       // 함수의 지역 변수
     {
      int i=10;  // 함수의 변수
      Print("Inside block i = ",i); // 결과는  i=10;
     }
   Print("Outside block i = ",i);  // 결과는  i=5;
  }
```

즉, 내부 블록이 실행되는 동안 외부 블록에 동일한 이름의 식별자 값이 아니라 자체 로컬 식별자 값이 표시됩니다.

예제:

```
void OnStart()
  {
//---
   int i=5;       // 함수의 지역 변수
   for(int i=0;i<3;i++)
      Print("안에서 i  = ",i);
   Print("블록 밖에서 i  = ",i);
  }
/* 실행 결과
Inside for i = 0
Inside for i = 1
Inside for i = 2
```

© 2000-2025, MetaQuotes Ltd.

```
블록 밖에서 i = 5
*/
```

정적 으로 선언된 로컬 변수는 프로그램 시작 이후 존재함에도 불구하고 블록의 범위를 가집니다.

Stack

모든 MQL5 프로그램에서는 자동으로 생성된 로컬 함수 변수를 저장하기 위해 스택이라는 특수 메모리 영역이 할당됩니다. 스택 하나가 모든 기능에 할당되며 지표에 대한 그것의 기본 크기는 1Mb입니다. Expert Advisors 및 스크립트에서는 스택 크기를 #property stacksize 컴파일러 명령문(바이트 단위로 스택 크기를 설정)을 사용하여 스택 크기를 관리할 수 있으며, 기본적으로 스택에 8Mb의 메모리가 할당됩니다.

정적 로컬 변수는 다른 정적 & 글로벌 변수가 저장되는 동일한 위치(스택과는 별도로 존재하는 특수 메모리 영역)에 저장됩니다. 동적으로 생성된 변수도 스택과 별도의 메모리 영역을 사용합니다.

각 함수 호출을 통해 스택의 위치가 내부 비정적 변수에 할당됩니다. 기능을 종료하면 메모리를 다시 사용할 수 있습니다.

첫 번째 함수에서 두 번째 함수를 호출하면 두 번째 함수는 해당 변수에 대해 나머지 스택 메모리에서 필요한 크기를 차지합니다. 따라서 포함된 함수를 사용할 때 각 기능에 대해 스택 메모리가 순차적으로 사용됩니다. 이로 인해 함수 호출 중 하나에서 메모리가 부족해질 수 있으며, 이를 스택 오버플로라고 합니다.

따라서 대용량 로컬 데이터의 경우 동적 메모리를 사용하는 것이 좋습니다. 함수를 입력할 때는 로컬 요구에 필요한 메모리를 시스템(new, ArrayResize())에 할당하고 기능을 종료할 때는 메모리(delete, ArrayFree())를 해제합니다.

더 보기

데이터 유형, 유형의 캡슐화 및 확장성,변수 초기화, 가시성 범위 및 변수 수명, 개체 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

형식 매개변수

함수에 전달되는 매개 변수는 local입니다. 범위는 함수 블럭입니다 형식 매개 변수는 외부 변수 및 한 함수 내에서 정의된 로컬 변수와 다른 이름을 가져야 합니다. 일부 값은 함수 블록의 형식 파라미터에 할당할 수 있습니다. 형식 매개 변수를 const 한정자로 선언하면 함수 내에서 값을 변경할 수 없습니다.

예제:

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
   ...
  }
```

형식 매개변수는 상수로 초기화될 수 있습니다. 이 경우 초기화 값이 기본값으로 간주됩니다. 초기화된 매개변수 옆에 있는 매개변수도 초기화해야 합니다.

예제:

```
void func(int x, double y = 0.0, bool z = true)
  {
   ...
  }
```

이러한 함수를 호출할 때 초기화된 매개 변수를 생략할 수 있으며 기본값 대신 대체됩니다.

예제:

```
func(123, 0.5);
```

단순 유형 매개 변수는 값별로 전달됩니다. 즉, 호출된 함수 내에서 이 유형의 해당 로컬 변수의 수정 사항이 호출 함수에 반영되지 않습니다. 모든 유형의 배열과 구조물 유형의 데이터는 항상 참조를 통해 전달됩니다. 배열 또는 구조 컨텐츠 수정을 금지해야 하는 경우 이러한 유형의 매개 변수는 *const* 키워드로 선언해야 합니다.

단순 유형의 매개변수를 레퍼런스로 전달할 수 있는 기회가 있습니다. 이 경우 호출 함수 내에서 이러한 매개변수를 수정하면 레퍼런스로 전달되는 해당 변수에 영향을 줍니다. 매개변수가 참조를 통과했음을 나타내려면 데이터 유형 뒤에 & 한정자를 넣습니다.

예제:

```
void func(int& x, double& y, double & z[])
  {
   double calculated_tp;
   ...
   for(int i=0; i<OrdersTotal(); i++)
     {
      if(i==ArraySize(z))        break;
      if(OrderSelect(i)==false) break;
      z[i]=OrderOpenPrice();
     }
   x=i;
```

© 2000-2025, MetaQuotes Ltd.

```
   y=calculated_tp;
  }
```

참조를 통해 전달된 매개 변수는 기본적으로 초기화할 수 없습니다.

함수에 최대 64개의 매개 변수를 전달할 수 있습니다.

더 보기

입력 변수, 데이터 유형, 유형의 캡슐화 및 확장성, 변수 초기화, 가시성 범위 및 변수 수명, 객체 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

정적 변수

static의 스토리지 클래스는 정적 변수를 정의합니다. 정적 한정자는 데이터 유형 앞에 표시됩니다.

예제:

```
int somefunc()
  {
   static int flag=10;
   ...
   return(flag);
  }
```

정적 변수는 모든 식으로 초기화할 수 있는 단순 로컬 변수와 달리 유형에 해당하는 상수 또는 상수 식에 의해 초기화될 수 있습니다.

정적 변수는 프로그램 실행 순간부터 존재하며 특수 함수 OnInit()가 호출되기 전에 한 번만 초기화됩니다. 초기 값을 지정하지 않으면 정적 저장소 클래스의 변수가 0의 초기 값을 가져갑니다.

로컬 변수 - *정적* 키워드로 선언된 -는 함수의 수명 동안 해당 값을 유지합니다. 각 다음 함수 호출에서 이러한 로컬 변수는 이전 호출 중에 가졌던 값을 포함합니다.

함수의 형식 매개 변수를 제외한 블록의 모든 변수는 정적으로 정의할 수 있습니다. 로컬 수준에서 선언된 변수가 정적 변수가 아닌 경우 해당 변수에 대한 메모리가 프로그램 스택에 자동으로 할당됩니다.

예제:

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0) Print("함수 카운터는 다음으로 불립니다 ",count," times");
   return count;
  }
void OnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
     {
      int c=Counter();
     }
   Print("c =",c);
  }
```

더 보기

데이터 유형, 유형의 캡슐화 및 확장성, 변수 초기화, 가시성 범위 및 변수 수명, 객체 생성 및 삭제, 정적 클래스 멤버

© 2000-2025, MetaQuotes Ltd.

글로벌 변수

글로벌 변수는 선언문을 함수 설명 외부에 배치하여 생성됩니다. 전역 변수는 함수와 동일한 수준에서 정의됩니다. 즉, 어떤 블록에서도 로컬 변수가 아닙니다.

예제:

```
int GlobalFlag=10;   // 전역 변수
int OnStart()
  {
   ...
  }
```

글로벌 변수의 범위는 전체 프로그램입니다. 전역 변수는 프로그램에 정의된 모든 함수에서 액세스할 수 있습니다. 다른 초기 값이 명시적으로 정의되지 않는 한 이 값은 0으로 초기화됩니다. 전역 변수는 해당 유형에 해당하는 상수 또는 상수 식을 통해서만 초기화할 수 있습니다.

글로벌 변수는 프로그램이 클라이언트 터미널 메모리에 로드된 후 및 Init 이벤트가 처음 처리되기 전에 한 번만 초기화됩니다. 클래스 개체를 나타내는 전역 변수의 경우 초기화 중에 해당 생성자를 호출합니다. 스크립트에서 전역 변수는 Start 이벤트를 처리하기 전에 초기화됩니다.

참고: 변수들 글로벌 수준으로 선언은 GlobalVariable...() 기능을 사용하여 액세스 할 수 있는 클라이언트 단말기 글로벌 변수와 함께 혼동되지 않아야 합니다.

더 보기

데이터 유형, 유형의 캡슐화 및 확장성,변수 초기화, 가시성 범위 및 변수 수명, 개체 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

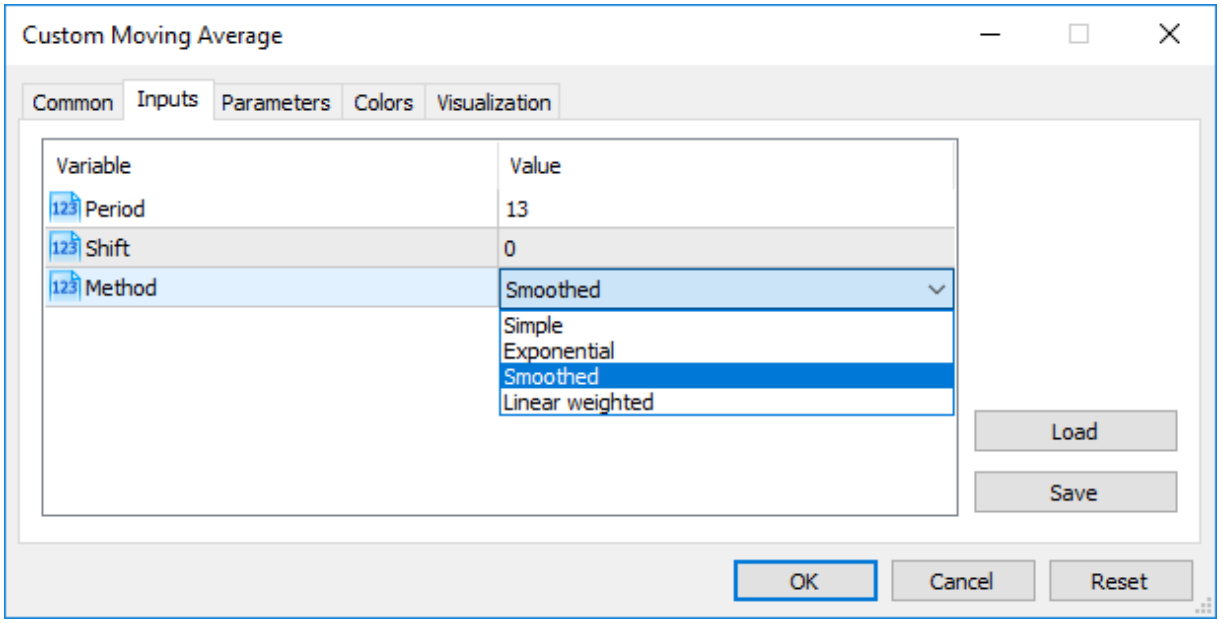
입력 변수

input 스토리지 클래스는 외부 변수를 정의합니다. *input* 한정자는 데이터 유형 앞에 표시됩니다. 입력 한정자가 있는 변수는 mql5 프로그램 내에서 변경할 수 없습니다. 이러한 변수는 읽기전용으로만 액세스할 수 있습니다. 입력 변수의 값은 프로그램 속성 창의 사용자만 변경할 수 있습니다. 외부 변수는 항상 OnInit()가 호출되기 직전에 다시 초기화됩니다.

예제:

```
//--- 입력 매개 변수
input int             MA_Period=13;
input int             MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

입력 변수는 프로그램의 입력 매개 변수를 결정합니다. 프로그램의 속성 창에서 사용할 수 있습니다.

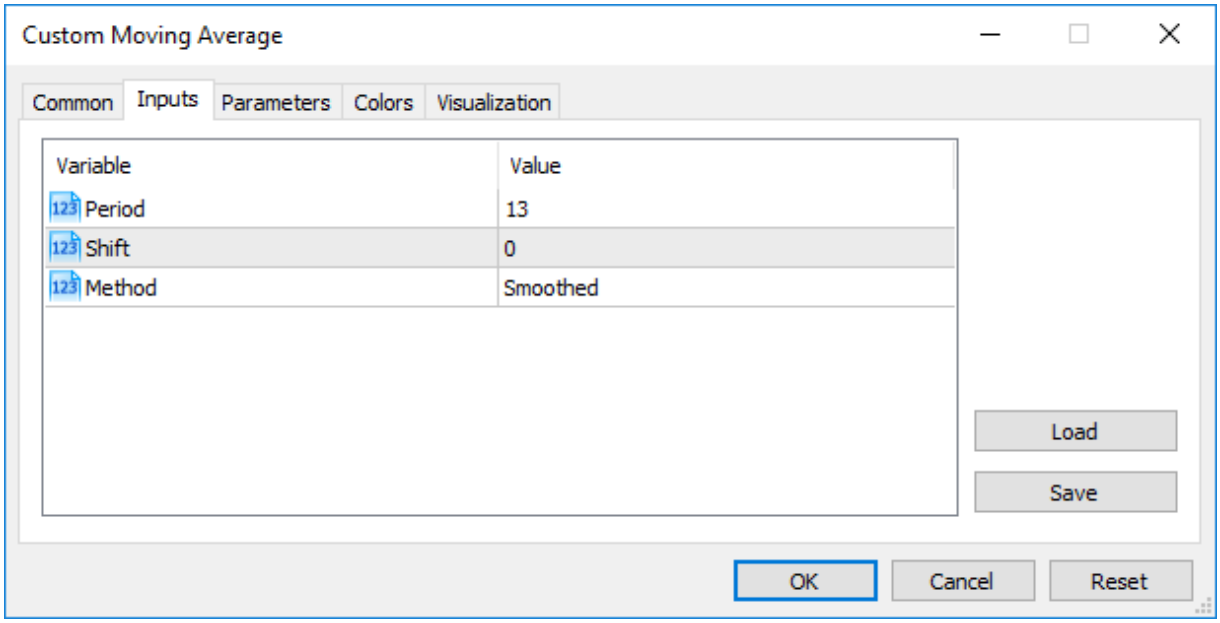


입력 탭에서 입력 파라미터의 모양을 설정하는 다른 방법이 있습니다. 이를 위해 입력 매개 변수의 설명 뒤에 같은 줄에 문자열 주석을 배치합니다. 이렇게 하면 입력 매개 변수의 이름을 사용자가 더 쉽게 이해할 수 있습니다.

예제:

```
//--- 입력 매개 변수
input int             InpMAPeriod=13;          // 평활 기간
input int             InpMAShift=0;            // 라인 수평 이동
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // 평활법
```

© 2000-2025, MetaQuotes Ltd.



참고: 복합 유형의 배열 및 변수는 입력 변수로 작동할 수 없습니다.

참고: 입력 변수에 대한 문자열 주석 길이는 63자를 초과할 수 없습니다.

MQL5 프로그램에서 커스텀 지표를 호출할 때 매개 변수 전달

커스텀 지표는 iCustom() 함수를 사용하여 호출됩니다. 커스텀 지표의 이름 뒤에는 매개 변수가 이 커스텀 지표의 입력 변수 선언에 엄격하게 따라야 합니다. 표시된 매개 변수가 호출된 커스텀 지표에 선언된 입력 변수보다 작으면 누락된 매개 변수는 변수 선언 중에 지정된 값으로 채워집니다.

커스텀 지표가 첫 번째 유형의 OnCalculate 함수를 사용하는 경우(즉, 지표가 동일한 데이터 배열을 사용하여 계산됨), 이러한 커스텀 지표를 호출할 때 다른 지표의 ENUM_APPLIED_PRICE 값 또는 핸들 중 하나를 마지막 매개변수로 사용해야 합니다. 입력 변수에 해당하는 모든 매개변수를 명확히 표시해야 합니다.

입력 매개 변수로서의 열거값

MQL5에서 제공하는 내장 열거뿐 아니라 커스텀 변수(mql5 프로그램의 입력 매개 변수)도 입력 변수로 사용할 수 있습니다. 예를 들어, 요일을 설명하는 dayOfWeek 열거를 생성하고 입력 변수를 사용하여 숫자가 아닌 일반적인 방법으로 특정 요일을 지정할 수 있습니다.

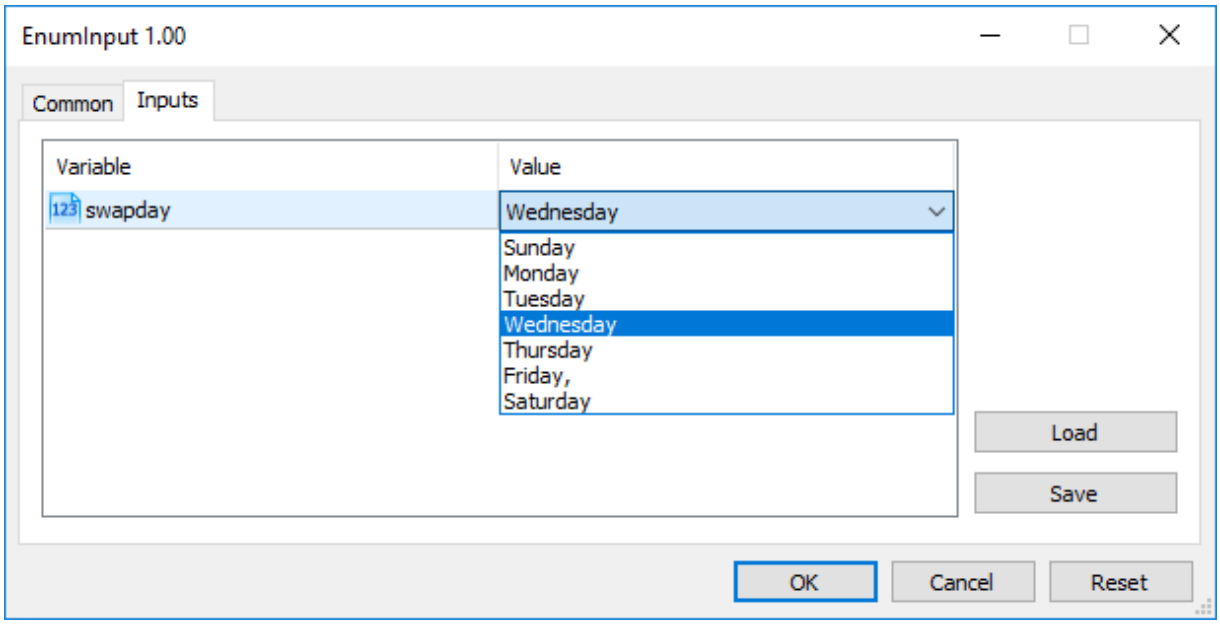
예제:

```
#property script_show_inputs
//--- 요일
enum dayOfWeek
  {
   S=0,      // 일요일
   M=1,      // 월요일
   T=2,      // 화요일
   W=3,      // 수요일
   Th=4,     // 목요일
   Fr=5,     // 금요일,
```

© 2000-2025, MetaQuotes Ltd.

```
   St=6,     // 토요일
  };
//--- 입력 매개 변수
input dayOfWeek swapday=W;
```

스크립트 시작 중에 사용자가 속성 창에서 필요한 값을 선택할 수 있도록 전처리기 명령 #property script_show_inputs 를 사용합니다. 스크립트를 시작하고 목록에서 dayOfWeek 열거값 중 하나를 선택할 수 있습니다. EnumInInput 스크립트를 시작하고 Inputs 탭으로 이동합니다. 기본적으로 스왑일 값(트리플 스왑 비용 청구일)은 수요일(W = 3)이지만 다른 값을 지정할 수 있으며 이 값을 사용하여 프로그램 작업을 변경할 수 있습니다.



열거 가능한 값 수가 제한됩니다. 입력 값을 선택하기 위해 드롭다운 목록이 사용됩니다. 열거형 멤버의 니모닉 이름은 목록에 표시된 값에 사용됩니다. 이 예와 같이 설명이 니모닉 이름과 연결된 경우 니모닉 이름 대신 주석 내용이 사용됩니다.

dayOfWeek 열거의 각 값은 0에서 6 사이의 값을 가지지만 매개 변수 목록에는 각 값에 대해 지정된 설명이 표시됩니다. 이를 통해 입력 매개변수에 대한 명확한 설명이 있는 프로그램을 작성할 수 있습니다.

sinput 한정자가 있는 변수

input 한정자를 가진 변수는 프로그램을 시작할 때 외부 매개변수 값을 설정할 수 있을 뿐만 아니라 전략 테스터에서 거래 전략을 최적화할 때도 필요합니다. 문자열 유형 중 하나를 제외한 각 입력 변수를 최적화에 사용할 수 있습니다.

때로는 테스터의 모든 통과 영역에서 일부 외부 프로그램 매개변수를 제외해야 합니다. sinput 메모리 한정자가 이러한 경우에 도입되었습니다. sinput은 정적 외부 변수 선언(input = 정적 입력)을 나타냅니다. 이는 Expert Advisor 코드에 다음과 같은 선언이 있음을 의미합니다

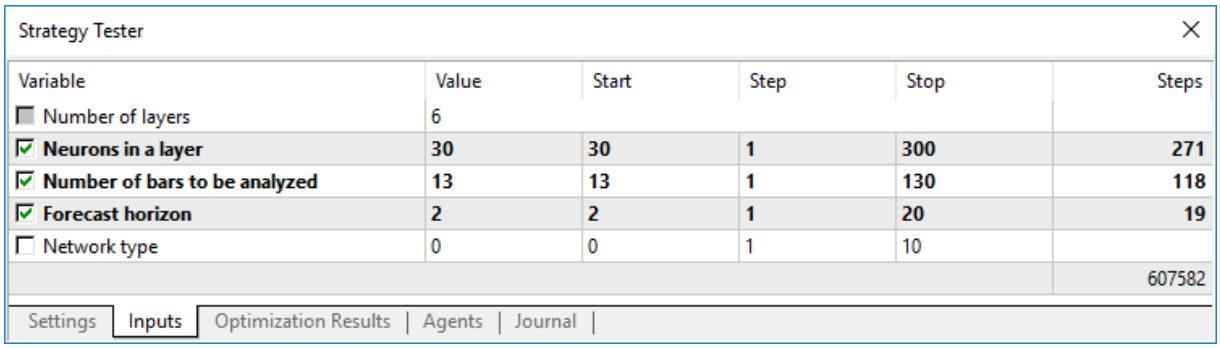
```
sinput       int layers=6;   // 레이어의 수
```

완전 선언과 동일할 것입니다

```
static input int layers=6;   // 레이어의 수
```

© 2000-2025, MetaQuotes Ltd.

sinput 한정자로 선언된 변수는 MQL5 프로그램의 입력 매개 변수입니다. 프로그램 실행 시 이 파라미터의 값을 변경할 수 있습니다. 그러나 이 변수는 입력 매개 변수의 최적화에는 사용되지 않습니다. 즉, 지정된 조건에 가장 적합한 매개변수 집합을 검색할 때 해당 값이 열거되지 않습니다.



위에 표시된 Expert Advisor에는 5가지 외부 매개 변수가 있습니다. "레이어 수" 는 sinput로 선언되고 6과 같습니다. 거래 전략 최적화 중에는 이 매개 변수를 변경할 수 없습니다. 그것이 더 이상 사용되는 데 필요한 값을 지정할 수 있습니다. 이러한 변수에는 시작, 단계 및 중지 필드를 사용할 수 없습니다.

따라서 변수에 대한 입력 수정자를 지정한 후에는 사용자가 이 매개 변수를 최적화할 수 없습니다. 즉, 터미널 사용자는 최적화 중에 지정된 범위의 자동 열거를 위해 전략 테스터에서 터미널의 초기 및 최종 값을 설정할 수 없습니다.

그러나 이 규칙에는 ParameterSetRange() 함수를 사용하여 최적화 작업에서 sinput 변수를 변경할 수 있다는 한 가지 예외가 있습니다. 이 함수는 모든 input 변수에 대해 사용 가능한 값 세트를 프로그램 제어하기 위해 특별히 도입 되었으며, 정적 인풋(sinput)으로 선언된 변수를 포함합니다. ParameterGetRange() 함수를 사용하면 최적화가 실행될 때(OnTesterInit() 핸들러에서) 입력 변수 값을 수신하고, 최적화된 매개변수 값이 열거될 변경 단계 값과 범위를 재설정할 수 있습니다.

이러한 방식으로 입력 매개 변수와 함께 작동하는 두 함수를 결합하면 다른 입력 매개 변수의 값에 따라 달라지는 입력 매개 변수의 최적화 간격을 설정하는 유연한 규칙을 만들 수 있습니다.

입력 매개 변수 정렬

MQL5 프로그램으로 편리하게 작업할 수 있도록 입력 매개변수를 group 키워드를 사용하여 명명된 블록으로 나눌 수 있습니다. 이를 통해 일부 매개 변수를 다른 매개 변수와 시각적으로 구분할 수 있습니다.

```
input group           "그룹명"
input int             variable1 = ...
input double          variable2 = ...
input diuble          variable3= ...
```

이러한 선언 후 차트 또는 전략 테스터에서 실행할 때 MQL5 사용자에 대한 매개 변수 구성을 단순화하는 모든 입력 매개 변수가 지정된 그룹에 시각적으로 결합됩니다. 새 그룹 선언이 나타날 때까지 각 그룹의 규격이 유효합니다:

```
input group           "그룹명 #1"
input int             group1_var1 = ...
input double          group1_var2 = ...
input double          group1_var3 = ...
```

© 2000-2025, MetaQuotes Ltd.

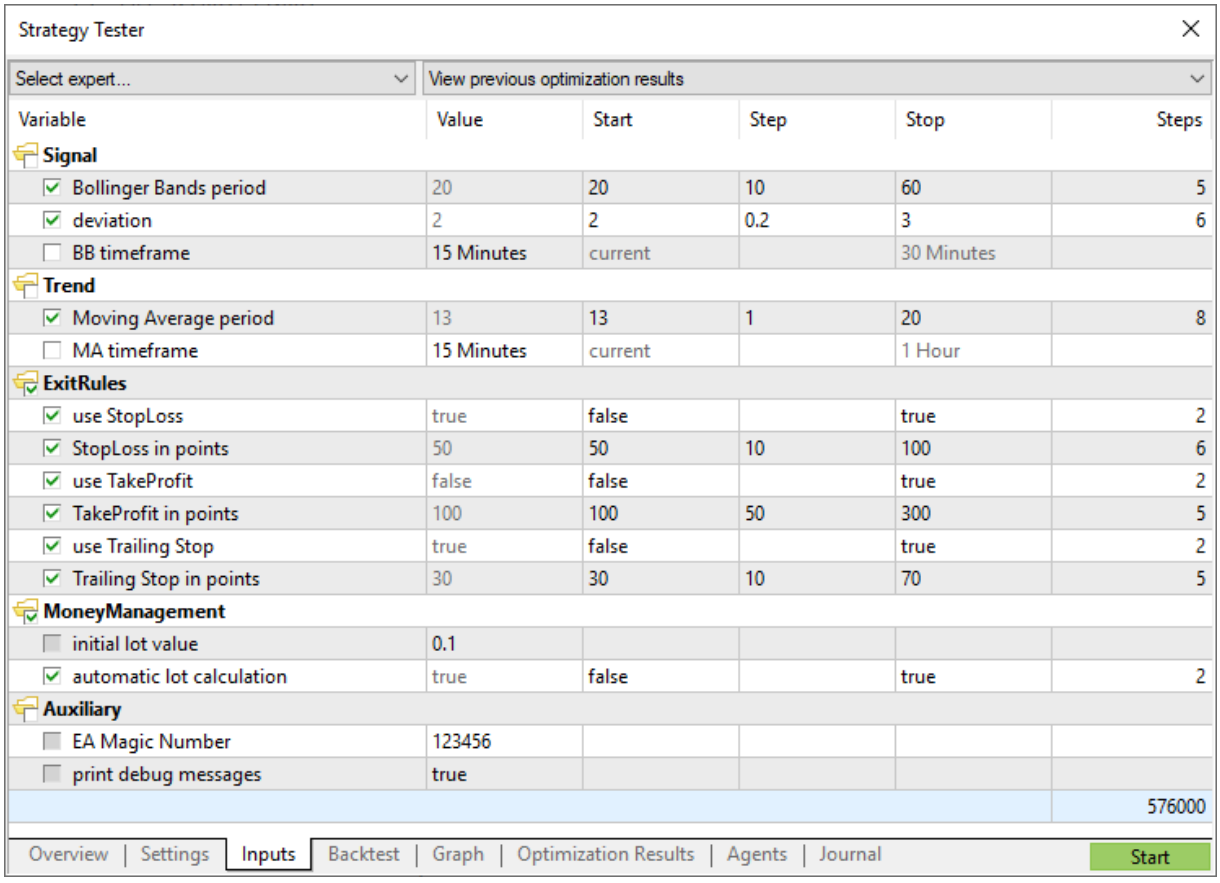
```
input group           "그룹명 #2
input int             group2_var1 = ...
input double          group2_var2 = ...
input double          group2_var3 = ...
```

입력 블록을 용도에 따라 구분한 샘플 EA:

```
input group           "시그널"
input int             ExtBBPeriod   = 20;       // 볼린저 밴드 기간
input double          ExtBBDeviation= 2.0;      // 편차
input ENUM_TIMEFRAMES ExtSignalTF=PERIOD_M15;   // BB 타임프레임

input group           "추세"
input int             ExtMAPeriod   = 13;       // 이동평균기간
input ENUM_TIMEFRAMES ExtTrendTF=PERIOD_M15;    // MA 타임프레임

input group           "ExitRules"
input bool            ExtUseSL      = true;     // StopLoss 사용
input int             Ext_SL_Points = 50;       // 포인트에서 StopLoss
input bool            ExtUseTP      = false;    // TakeProfit 사용
input int             Ext_TP_Points = 100;      // 포인트에서 TakeProfit
input bool            ExtUseTS      = true;     // Trailing Stop 사용
input int             Ext_TS_Points = 30;       // 포인트에서 Trailing Stop

input group           "MoneyManagement"
sinput double         ExtInitialLot = 0.1;      // 초기 롯 값
input bool            ExtUseAutoLot = true;     // 자동 롯 계산

input group           "보조적"
sinput int            ExtMagicNumber = 123456;  // EA 매직 넘버
sinput bool           ExtDebugMessage= true;    // 디버그 메시지 출력
```

전략 테스터에서 이러한 EA를 실행할 때 그룹 이름을 두 번 클릭하여 입력 블록을 축소/확장할 수 있으며 그룹 확인란을 클릭하여 최적화를 위한 모든 매개 변수를 선택할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.



더 보기

iCustom, 열거, 프로그램 속성

© 2000-2025, MetaQuotes Ltd.

외부 변수

extern 키워드는 변수 식별자를 정적 스토리지 클래스 – 전역 lifetime을 가진 – 식별자로 선언하는 데 사용됩니다. 이 변수들은 프로그램 시작부터 존재하며 프로그램 시작 후 즉시 메모리를 할당하고 초기화합니다.

여러 원본 파일로 구성된 프로그램을 만들 수 있습니다. 이 경우에는 전처리기 #include에 대한 지시어가 사용됩니다. 유형 및 식별자가 동일한 외부 변수로 선언된 변수는 한 프로젝트의 서로 다른 원본 파일에 존재할 수 있습니다.

전체 프로젝트를 컴파일할 때 유형과 식별자가 동일한 모든 외부 변수가 글로벌 변수 풀의 메모리 중 하나와 연결됩니다. 외부 변수는 원본 파일을 별도로 컴파일하는 데 유용합니다. 외부 변수는 초기화할 수 있지만 한 번만 – 동일한 유형 및 동일한 식별자로 초기화된 여러 외부 변수가 존재할 수 없습니다.

더 보기

데이터 유형, 유형의 캡슐화 및 확장성, 변수 초기화, 가시성 범위 및 변수 수명, 개체 생성 및 삭제

© 2000–2025, MetaQuotes Ltd.

변수 초기화

정의하는 동안 모든 변수를 초기화할 수 있습니다. 변수가 명시적으로 초기화되지 않은 경우 이 변수에 저장된 값은 임의의 값이 될 수 있습니다. 암시적 초기화가 사용되지 않습니다.

전역 및 정적 변수는 해당 유형의 상수 또는 상수 식을 통해서만 초기화할 수 있습니다. 로컬 변수는 상수가 아닌 임의의 식을 사용하여 초기화할 수 있습니다.

전역 및 정적 변수의 초기화는 한 번만 수행됩니다. 해당 함수를 호출할 때마다 로컬 변수가 초기화됩니다.

예제:

```
int    n       = 1;
string s       = "hello";
double f[]     = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4] = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4 } };
//--- 테트리스에서
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- 값이 0인 구조의 모든 필드 초기화
MqlTradeRequest request={0};
```

배열 요소의 값 목록은 대괄호로 묶어야 합니다. 누락된 초기화 시퀀스는 0으로 간주됩니다. 초기화 시퀀스에는 하나 이상의 값이 있어야 합니다. 이 값은 해당 구조 또는 배열의 첫 번째 요소로 초기화되고 누락된 요소는 0과 같은 것으로 간주됩니다.

초기화된 배열의 크기를 지정하지 않으면 초기화 시퀀스의 크기에 따라 컴파일러에 의해 결정됩니다. 1차원 배열은 하나의 초기화 요소(0, 규칙)만 지정된 경우를 제외하고 1차원 시퀀스(추가 곱슬괄호가 없는 시퀀스)로 초기화할 수 없습니다.

로컬 수준에서 발표된 배열을 포함한 배열은 상수로만 초기화할 수 있습니다.

예제:

```
struct str3
  {
   int               low_part;
   int               high_part;
  };
struct str10
  {
   str3              s3;
   double            d1[10];
   int               i3;
  };
void OnStart()
  {
   str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
   str10 s10_2={{1,0},{0},100};
   str10 s10_3={{1,0},{1.0}};
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
   Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
   Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
   Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
  }
```

구조 유형 변수의 경우 정적 배열(암묵적으로 설정된 크기)뿐만 아니라 부분 초기화가 허용됩니다. 구조 또는 배열의 첫 번째 요소를 하나 이상 초기화할 수 있으며, 이 경우 다른 요소는 0으로 초기화됩니다.

더 보기

데이터 유형, 유형의 캡슐화 및 확장성, 가시성 범위 및 변수 수명, 객체 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

가시성 범위 및 변수 수명

범위에는 두 가지 기본 유형이 있습니다: 로컬 범위 및 글로벌 범위.

모든 함수 외부에 선언된 변수는 글로벌 범위에 있습니다. 이러한 변수에 대한 액세스는 프로그램의 어디에서나 수행할 수 있습니다.이러한 변수는 글로벌 메모리 풀에 있으므로 프로그램 수명과 일치합니다.

블록 내부에 선언된 변수(대괄호로 묶인 코드의 일부)는 로컬 범위에 속합니다. 이러한 변수는 선언된 블록 외부에 표시되지 않으므로 사용할 수 없습니다. 로컬 선언의 가장 일반적인 경우는 함수 내에서 선언된 변수입니다. 로컬로 선언된 변수는 스택에 있으며 이러한 변수의 수명은 함수의 수명과 같습니다.

로컬 변수의 범위는 선언된 블록이기 때문에 다른 블록에서 선언된 변수뿐만 아니라 상위 수준에서 선언된 변수와 이름이 같은 변수를 글로벌 수준까지 선언할 수 있습니다.

예제:

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double &price[]
  {
   int          i,limit;
   static int weightsum=0;
   double       sum=0;
//---
   if(prev_calculated==0)
     {
      limit=MA_Period+begin;
      //--- 첫 번째 한계 막대의 빈 값 설정
      for(i=0; i<limit; i++) LineBuffer[i]=0.0;
      //--- 첫번째 가시적 값 계산
      double firstValue=0;
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
      firstValue/=(double)weightsum;
      LineBuffer[limit-1]=firstValue;
     }
   else
     {
      limit=prev_calculated-1;
     }

   for(i=limit;i<rates_total;i++)
     {
      sum=0;
      for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
      LineBuffer[i]=sum/weightsum;
     }
//---
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

라인에 선언된 변수 i에 주의하십시오

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

해당 범위는 for 루프입니다. 이 루프 밖에는 함수의 시작 부분에 선언된 동일한 이름의 다른 변수가 있습니다. 또한 k 변수는 루프 본문에 선언되며, 그 범위는 루프 본문입니다.

로컬 변수는 액세스 지정자 static을 사용하여 선언할 수 있습니다. 이 경우 컴파일러는 글로벌 메모리 풀에 변수를 가집니다. 따라서 정적 변수의 수명은 프로그램의 수명과 같습니다. 여기서 이러한 변수의 범위는 선언된 블록으로 제한됩니다.

더 보기

데이터 유형, 유형의 캡슐화 및 확장성,변수 초기화, 객체 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

객체 생성 및 삭제

실행을 위해 MQL5 프로그램을 로드한 후에는 유형에 따라 각 변수에 메모리가 할당됩니다. 액세스 수준에 따라 모든 변수는 글로벌 변수 및 로컬 변수의 두가지 유형으로 나뉩니다. 메모리 클래스에 따라 MQL5 프로그램의 입력 매개변수가 될 수 있습니다. 정적 및 자동으로. 필요한 경우 각 변수는 해당 값으로 초기화됩니다. 변수를 사용한 후 변수가 초기화되지 않고 사용된 메모리가 MQL5 실행 파일 시스템으로 반환됩니다.

전역 변수의 초기화 및 초기화 취소

전역 변수는 MQL5 프로그램이 로드된 직후와 함수가 호출되기 전에 자동으로 초기화됩니다. 초기화 중에 초기 값은 단순 유형의 변수에 할당되고 생성자(있는 경우)가 객체에 대해 호출됩니다. 입력 변수는 항상 글로벌 수준에서 선언되며 프로그램 시작 중에 대화 상자에서 사용자가 설정한 값으로 초기화됩니다.

일반적으로 정적 변수가 로컬 수준에서 선언되지만 이러한 변수에 대한 메모리는 사전 할당되며 글로벌 변수와 마찬가지로 프로그램이 로드된 직후 초기화가 수행됩니다.

초기화 순서는 프로그램의 변수 선언 순서에 해당합니다. 초기화 취소는 역순으로 수행됩니다. 이 규칙은 새 연산자가 생성하지 않은 변수에 대해서만 적용됩니다. 이러한 변수는 로드 직후 자동으로 생성 및 초기화되며 프로그램 언로드 전에 초기화 취소됩니다.

로컬 변수의 초기화 및 초기화 취소

로컬 수준에서 선언된 변수가 정적 변수가 아닌 경우 해당 변수에 대해 메모리가 자동으로 할당됩니다. 로컬 변수와 전역 변수는 프로그램 실행이 선언을 충족하는 시점에 자동으로 초기화됩니다. 따라서 초기화 순서는 선언 순서에 해당합니다.

로컬 변수는 선언된 프로그램 블록의 끝에서 선언과 반대 순서로 초기화됩니다. 프로그램 블록은 복합 연산자인데, 이는 다음 선택 연산자들의 부분이 될 수 있습니다: switch, 루프 연산자 (for, while, do-while), 함수 바디: if-else 연산자.

로컬 변수는 프로그램 실행이 변수 선언을 충족하는 순간에만 초기화됩니다. 프로그램 실행 중에 변수가 선언된 블록이 실행되지 않은 경우 이러한 변수는 초기화되지 않습니다.

배치된 개체의 초기화 및 초기화 해제

포인터 선언에는 해당 개체의 초기화가 수반되지 않기 때문에 객체 포인터를 사용하는 경우가 특별합니다. 동적으로 배치된 객체는 클래스 샘플이 new 연산자에 의해 생성되는 순간에만 초기화됩니다. 객체의 초기화는 해당 클래스의 생성자의 호출을 전제로 합니다. 클래스에 해당 생성자가 없으면 simple 유형의 멤버가 자동으로 초기화되지 않고 문자열, 동적 배열 및 복합 객체 유형의 멤버가 자동으로 초기화됩니다.

포인터는 로컬 또는 글로벌 수준에서 선언할 수 있으며 NULL의 빈 값 또는 동일하거나 또는 상속된 유형의 포인터 값으로 초기화할 수 있습니다. 로컬 수준에서 선언된 포인터에 대해 *new* 연산자를 호출하는 경우, 해당 수준을 종료하기 전에 이 포인터에 대한 *삭제* 연산자를 수행해야 합니다. 그렇지 않으면 포인터가 손실되고 개체의 명시적 삭제가 실패합니다.

object_pointer=new *Class_name* 표현식으로 작성된 모든 개체는 delete(*object_pointer*) 연산자에 의해 삭제되어야 합니다. 어떤 이유로 프로그램이 완료될 때 삭제 연산자가 이러한 변수를 삭제하지 않으면 해당 항목이 "Experts" 저널에 나타납니다. 여러 변수를 선언하고 한 개체의 포인터를 모든 변수에 할당할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

동적으로 작성된 객체에 생성자가 있는 경우 *new* 연산자 실행 시 이 생성자가 호출됩니다. 개체에 소멸자가 있는 경우 *delete* 연산자를 실행하는 동안 호출됩니다.

따라서 동적으로 배치된 객체는 해당 *new* 연산자가 호출되는 순간에만 생성되며 프로그램 언로드 중에 *delete* 연산자에 의해 또는 MQL5의 실행 시스템에 의해 자동으로 삭제됩니다. 동적으로 생성된 개체의 포인터 선언 순서는 초기화 순서에 영향을 주지 않습니다 초기화 및 초기화 해제 순서는 프로그래머가 완전히 제어합니다.

MQL5의 동적 메모리 할당

동적 배열로 작업할 때 릴리스된 메모리는 즉시 운영 체제로 돌아갑니다.

new 연산자를 사용하여 동적 클래스 개체로 작업할 때 메모리 관리자가 작업 중인 클래스 메모리 풀에서 첫 번째 메모리가 요청됩니다. 풀에 메모리가 충분하지 않으면 운영 체제에서 메모리가 요청됩니다. delete 연산자를 사용하여 동적 개체를 삭제하면 릴리스된 메모리가 클래스 메모리 풀로 즉시 반환됩니다 .

메모리 관리자는 다음 이벤트 처리 기능을 종료한 후 즉시 운영 체제로 메모리를 다시 해제합니다: OnInit(), OnDeinit(), OnStart(), OnTick(), OnCalculate(), OnTimer(), OnTrade(), OnTester(), OnTesterInit(), OnTesterPass(), OnTesterDeinit(), OnChartEvent(), OnBookEvent().

변수의 간략한 특성

생성, 삭제 순서, 생성자 및 소멸자 호출에 대한 주요 정보가 아래 표에 나와 있습니다.

	글로벌 자동 변수	로컬 자동 변수	동적으로 생성된 객체
초기화	mql5 프로그램이 로드된 직후	실행 중에 선언된 코드 행에 도달하는 경우	**new** 연산자 실행시
초기화 주문	선언 순서대로	선언 순서대로	선언 순서를 불문
초기화 해제	mql5 프로그램을 언로드하기 전에	선언 블록을 실행 종료할 때	**delete** 연산자가 실행되거나 mql5 프로그램이 언로드되기 전에
초기화 해제 순서	초기화 순서와 반대 순서로	초기화 순서와 반대 순서로	초기화 순서와 상관없이
생성자 호출	mql5 프로그램 로딩시	초기화시	*new* 연산자 실행시
소멸자 호출	mql5 프로그램 언로딩시	변수가 초기화된 블록을 종료할 때	*delete* 연산자 실행시
오류 로그	자동으로 작성된 오브젝트를 삭제하려는 시도에 대한 메시지를 "Experts" 저널에 기록합니다	자동으로 작성된 오브젝트를 삭제하려는 시도에 대한 메시지를 "Experts" 저널에 기록합니다	mql5 프로그램 언로드 시 동적으로 생성되지 않은 개체에 대한 메시지를 "Experts" 저널에 기록합니다

더 보기

데이터 유형, 유형의 캡슐화 및 확장성,변수 초기화, 가시성 범위 및 변수 수명

© 2000-2025, MetaQuotes Ltd.

프리프로세서

프리프로세서는 프로그램 컴파일 직전에 프로그램 소스 코드를 준비하기 위한 MQL5 컴파일러의 특수 하위 시스템입니다.

프리프로세서를 사용하면 소스 코드 가독성을 향상시킬 수 있습니다. 이 코드는 mql5-프로그램의 소스 코드가 포함된 특정 파일을 포함하여 구성할 수 있습니다. 특정 상수에 니모닉 이름을 할당할 수 있으므로 코드 가독성이 향상됩니다.

전처리 기능을 사용하면 mql5-프로그램의 특정 매개 변수를 확인할 수도 있습니다:

- 상수 선언
- 프로그램 속성 설정
- 프로그램 텍스트에 파일 포함
- 가져오기 함수
- 조건부 컴파일

프리프로세서 명령문은 컴파일하기 전에 소스 코드를 전처리하는 데 사용됩니다. 명령문은 항상 #로 시작하므로 컴파일러는 변수, 함수 등의 이름에 기호를 사용할 수 없습니다.

각 명령문은 별도의 항목으로 설명되며 줄 바꿈까지 유효합니다. 한 항목에는 여러 명령문을 사용할 수 없습니다. 지시어 입력이 너무 크면 ₩기호를 사용하여 여러 줄로 구분할 수 있습니다. 이 경우 다음 행은 명령문 입력의 연속으로 간주됩니다.

```
//+------------------------------------------------------------------+
//|  foreach pseudo-operator                                         |
//+------------------------------------------------------------------+
#define ForEach(index, array) for (int index = 0,                    \
   max_##index=ArraySize((array));                                   \
   index<max_##index; index++)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string array[]={"12","23","34","45"};
//--- For Each를 이용한 배열 우회
   ForEach(i,array)
     {
      PrintFormat("%d: array[%d]=%s",i,i,array[i]);
     }
  }
//+------------------------------------------------------------------+
/* 아웃풋 결과
   0: array[0]=12
   1: array[1]=23
   2: array[2]=34
   3: array[3]=45
*/
```

© 2000-2025, MetaQuotes Ltd.

컴파일러의 경우 이 세 가지 #define 명령문들이 모두 하나의 긴 줄처럼 보입니다. 위의 예제는 #definite 매크로에서 두 매크로 토큰을 하나로 병합하는 데 사용되는 병합 연산자 ## 문자도 적용합니다. 토큰 병합 연산자는 매크로 정의의 처음 또는 마지막 연산자가 될 수 없습니다.

© 2000-2025, MetaQuotes Ltd.

매크로 대체 (#define)

프리프로세서 명령문은 컴파일하기 전에 소스 코드를 전처리하는 데 사용됩니다. 명령문은 항상 #로 시작하므로 컴파일러는 변수, 함수 등의 이름에 기호를 사용할 수 없습니다.

각 명령문은 별도의 항목으로 설명되며 줄 바꿈까지 유효합니다. 한 항목에는 여러 명령문을 사용할 수 없습니다. 지시어 입력이 너무 크면 ₩기호를 사용하여 여러 줄로 구분할 수 있습니다. 이 경우 다음 행은 명령문 입력의 연속으로 간주됩니다.

#define 명령문을 사용하여 니모닉 이름을 상수에 할당할 수 있습니다. 두가지 형태가 있습니다:

```
#define identifier expression                    // 매개 변수가 없는 포맷
#define identifier(par1,... par8) expression     // 매개변수가 있는 포맷
```

#define 명령문은 다음을 **표현식**으로 대체합니다: 원본 텍스트에서 추가로 발견된 *식별자*의 모든 항목. *식별자*는 별도의 토큰인 경우에만 대체됩니다. *식별자*는 주석, 문자열 또는 다른 긴 식별자의 일부인 경우 대체되지 않습니다.

상수 식별자는 변수 이름과 동일한 규칙에 의해 제어됩니다. 값은 모든 유형이 될 수 있습니다:

```
#define ABC               100
#define PI                3.14
#define COMPANY_NAME      "MetaQuotes Software Corp."
...
void ShowCopyright()
  {
   Print("Copyright  2001-2009, ",COMPANY_NAME);
   Print("https://www.metaquotes.net");
  }
```

표현식은 키워드, 상수, 상수 및 비변환 식과 같은 여러 토큰으로 구성될 수 있습니다. **표현식**은 행의 끝으로 끝나며 다음 행으로 전송할 수 없습니다.

예제:

```
#define TWO         2
#define THREE       3
#define INCOMPLETE TWO+THREE
#define COMPLETE   (TWO+THREE)
void OnStart()
  {
   Print("2 + 3*2 = ",INCOMPLETE*2);
   Print("(2 + 3)*2 = ",COMPLETE*2);
  }
// 결과
// 2 + 3*2 = 8
// (2 + 3)*2 = 10
```

Parametric Form #define

© 2000-2025, MetaQuotes Ltd.

파라메트릭 형태를 사용하면 이후 발견된 모든 식별자 항목이 실제 매개변수를 고려한 식으로 대체됩니다. 예:

```
 // 두 개의 매개 변수가 a 및 b인 예제
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
expression double c=MUL(A,B);
다음과 같습니다 doubl e c=((2+3)*(5-1));
*/
// 결과
// c=20
```

식에서 매개 변수를 사용할 때는 매개 변수를 괄호로 묶어야 합니다. 이렇게 하면 쉽게 찾을 수 없는 오류를 방지할 수 있습니다. 대괄호를 사용하지 않고 코드를 다시 쓰면 결과가 달라집니다:

```
 // 두 개의 매개 변수가 a 및 b인 예제
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
expression double c=MUL(A,B);
다음과 같습니다 doubl e c=2+3*5-1;
*/
// 결과
// c=16
```

모수 형식을 사용할 경우 최대 8개의 모수가 허용됩니다.

```
 // 올바른 매개변수 형식
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)   // one parameter - 'text

 // 잘못된 매개변수 형식
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // p1에서 p9까지 8
```

#undef 명령문

#undef 명령문은 이전에 정의된 매크로 대체의 선언을 취소합니다.

예제:

```
#define MACRO

```

© 2000-2025, MetaQuotes Ltd.

```
void func1()
  {
#ifdef MACRO
   Print("MACRO는 다음에 정의되어 있습니다: ",__FUNCTION__);
#else
   Print("MACRO가 다음에 정의되어 있지 않습니다: ",__FUNCTION__);
#endif
  }

#undef MACRO

void func2()
  {
#ifdef MACRO
   Print("MACRO가 다음에 정의되어 있습니다: ",__FUNCTION__);
#else
   Print("MACRO가 다음에 정의되어 있지 않습니다: ",__FUNCTION__);
#endif
  }

void OnStart()
  {
   func1();
   func2();
  }

/* 결과:
MACRO가 func1에 정의되어 있습니다
MACRO가 func2에 정의되어 있지 않습니다
*/
```

더 보기

식별자, 문자 상수

© 2000-2025, MetaQuotes Ltd.

프로그램 속성 (#property)

모든 mql5-program에서는 명시적으로 프로그램을 실행할 필요 없이 프로그램에 대한 적절한 서비스 시 클라이언트 터미널에 도움이 되는 #property 라는 특정 매개 변수를 추가로 지정할 수 있습니다. 이는 우선 지표의 외부 설정과 관련이 있습니다. 포함된 파일에 설명된 속성은 완전히 무시됩니다. 기본 mq5 파일에 속성을 지정해야 합니다.

```
#property 식별자 값
```

컴파일러는 실행된 모듈의 구성에서 선언된 값을 씁니다.

상수	유형	설명
아이콘	string	EX5 프로그램의 아이콘으로 사용할 이미지 파일의 경로. 경로 지정 규칙은 리소스와 동일합니다. 기본 모듈에서 MQL5 소스 코드를 사용하여 속성을 지정해야 합니다. 아이콘 파일은 반드시 다음에 있어야합니다: ICO format.
링크	string	회사 웹 사이트 링크
저작권	string	회사명
버전	string	프로그램 버전, 최대 31자
설명	string	mql5-프로그램에 대한 간단한 텍스트 설명. 여러 개의 *설명*이 있을 수 있으며, 각 설명은 텍스트의 한 줄을 설명합니다. 모든 *설명*의 총 길이는 줄 피드를 포함하여 511자를 초과할 수 없습니다.
stacksize	int	MQL5 프로그램 스택 사이즈. 함수 재귀 호출을 실행할 때 충분한 크기의 스택이 필요합니다. 차트에서 스크립트 또는 Expert Advisor를 시작하면 최소 8MB의 스택이 할당됩니다. 지표의 경우 스택 크기는 항상 고정되고 1MB와 같습니다. 전략 테스터에서 프로그램이 실행되면 항상 16MB의 스택이 할당됩니다.
library		라이브러리, 시작 함수가 할당되지 않음, 익스포트 한정자가 있는 함수를 다른 mql5-프로그램에서 임포트 할 수 있음
indicator_applied_price	int	"적용 대상" 필드의 기본값을 지정합니다. ENUM_APPLIED_PRICE값 중 하나를 지정할 수 있습니다. 속성을 지정하지 않으면 기본값은 PRICE_CLOSE입니다
indicator_chart_window		차트 창에 지표 표시
indicator_separate_window		별도의 창에 지표 표시
indicator_height	int	픽셀 단위의 지표 하위 창의 고정 높이 (속성 INDICATOR_HEIGHT)
indicator_buffers	int	지표 계산을 위한 버퍼 수

© 2000-2025, MetaQuotes Ltd.

상수	유형	설명
indicator_plots	int	지표의 그래픽 시리즈 수
indicator_minimum	double	별도의 지표 창에 대한 하한 스케일링 한계
indicator_maximum	double	별도의 지표 창에 대한 상위 스케일링 한계
indicator_labelN	string	DataWindow에 표시되는 N번째 그래픽 시리즈의 레이블을 설정합니다. 여러 지표 버퍼(DRAW_CANDLES, DRAW_FILLING 등)가 필요한 그래픽 시리즈의 경우 레이블 이름은 구분 기호 ';'를 사용하여 정의됩니다.
indicator_colorN	색상	선 N을 표시하는 색상이며, 여기서 N은 그래픽 시리즈의 수입니다; 번호는 1부터 시작합니다
indicator_widthN	int	그래픽 시리즈의 선 두께, 여기서 N은 그래픽 시리즈의 수이며, 번호는 1부터 시작합니다.
indicator_styleN	int	그래픽 시리즈의 선 스타일, ENUM_LINE_STYLE 값으로 지정된. N은 그래픽 시리즈의 수입이며, 번호는 1부터 시작합니다.
indicator_typeN	int	ENUM_DRAW_TYPE 값으로 지정된 그래픽 플롯 유형. N은 그래픽 시리즈의 수입니다. 번호는 1부터 시작합니다
indicator_levelN	double	별도의 지표 창에서 N의 수평 수준
indicator_levelcolor	색상	지표의 수평 레벨 색상
indicator_levelwidth	int	지표의 수평 레벨 두께
indicator_levelstyle	int	지표의 수평 레벨 스타일
script_show_confirm		스크립트를 실행하기 전에 확인 창 표시
script_show_inputs		스크립트를 실행하기 전에 속성이 있는 창을 표시하고 이 확인 창을 사용하지 않도록 설정합니다
tester_indicator	string	"*indicator_name.ex5*" 형식의 커스텀 지표 이름입니다. 테스트가 필요한 지표는 해당 파라미터가 상수 문자열을 통해 설정된 경우 iCustom() 함수의 호출에서 자동으로 정의됩니다. 다른 모든 경우(IndicatorCreate() 함수 사용 또는 지표 이름을 설정하는 매개 변수에서 일정하지 않은 문자열 사용) 이 속성이 필요합니다
tester_file	string	확장자를 나타내는 테스터의 파일 이름(상수 문자열로 큰따옴표)입니다. 지정한 파일이 테스터로 전달됩니다. 필요한 입력 파일이 있는 경우 테스트할 입력 파일을 항상 지정해야 합니다.
tester_library	string	확장자가 큰따옴표로 표시된 라이브러리 이름. 라이브러리는 파일 확장자로 'dll' 또는 'ex5'를 사용할 수 있습니다. 테스트가 필요한 라이브러리는 자동으로

© 2000-2025, MetaQuotes Ltd.

상수	유형	설명
		정의됩니다. 그러나 커스텀 지표에 사용되는 라이브러리가 있는 경우 이 속성이 필요합니다
tester_set	string	입력 매개 변수의 단계와 값이 포함된 세트 파일의 이름. 테스트 및 최적화 전에 파일이 테스터에 전달됩니다. 파일 이름은 확장명과 큰따옴표로 상수 문자열로 지정됩니다. 설정된 파일 이름에 EA 이름 및 버전 번호를 "<expert_name>_<number>.set" 으로 지정하면, <number> 버전 번호 아래에 매개변수 버전 다운로드 메뉴에 자동으로 추가됩니다. 예를 들어 "MACD Sample_4.set"라는 이름은 버전 번호가 4인 "MACD Sample.mq5" EA의 세트 파일임을 의미합니다. 포맷을 연구하려면 테스트/최적화 설정을 전략 테스터에 수동으로 저장한 다음 이 방법으로 만든 세트 파일을 여는 것이 좋습니다.
tester_no_cache	string	최적화를 수행할 때 전략 테스터는 실행된 패스의 모든 결과를 최적화 캐시에 저장하며, 여기서 입력 매개변수의 각 세트에 대한 테스트 결과가 저장됩니다. 따라서 재계산 시간을 낭비하지 않고 동일한 매개변수로 재최적화하는 동안 이미 만들어진 결과를 사용할 수 있습니다. 그러나 일부 작업(예: 산술 계산)에서는 최적화 캐시에서 이미 만들어진 결과를 사용할 수 있는지에 관계없이 계산을 수행해야 할 수도 있습니다. 이 경우 파일에 *tester_no_cache* 속성이 포함되어야 합니다. 테스트 결과는 여전히 캐시에 저장되므로 수행된 통과에 대한 모든 데이터를 전략 테스터에서 볼 수 있습니다.
tester_everytick_calculate	string	Strategy Tester에서 지표는 데이터에 액세스할 때, 즉 지표 버퍼 값이 요청될 때만 계산됩니다. 이렇게 하면 각 눈금에 대한 지시자 값을 얻을 필요가 없는 경우 훨씬 더 빠른 검정 및 최적화 속도를 얻을 수 있습니다. *tester_everytick_calculate* 특성을 지정하면 모든 틱에서 지시자의 강제 계산을 활성화할 수 있습니다. 전략 테스터의 지표는 다음과 같은 경우에도 모든 틱에 대해 강제로 계산됩니다: • 시각적 모드에서 시험할 때; • 지표에 다음 함수가 있는 경우: EventChartCustom, OnChartEvent, OnTimer;

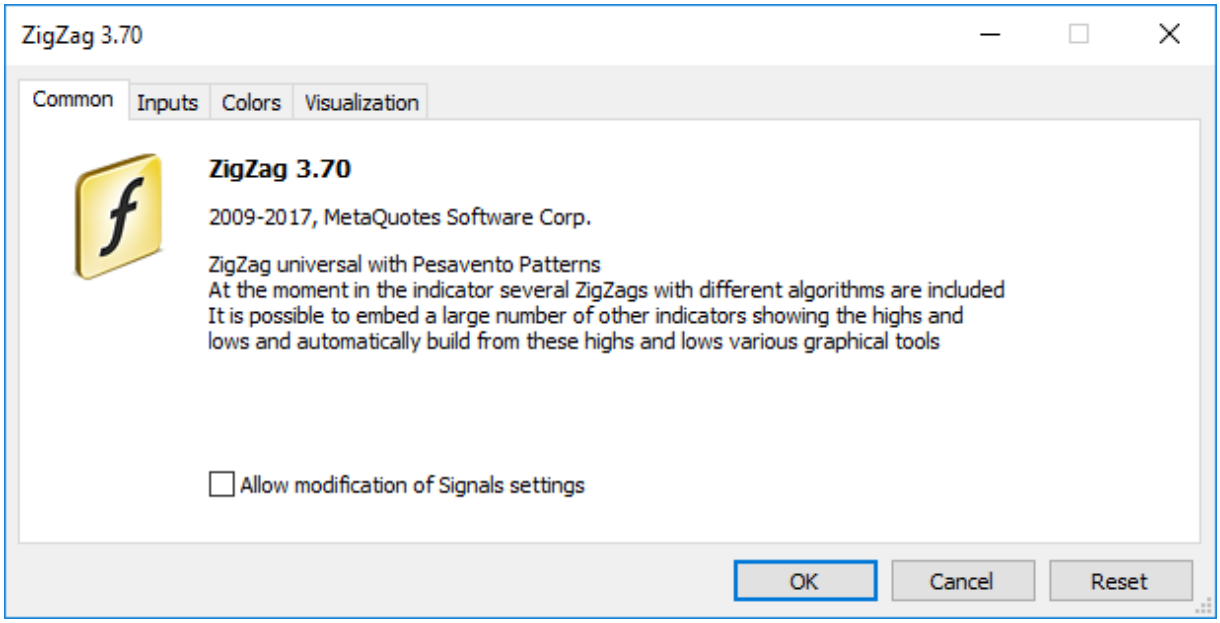
© 2000-2025, MetaQuotes Ltd.

상수	유형	설명
		• 지표가 1916 아래의 빌드 넘버로 컴파일러를 사용하여 생성되었는지 여부. 이 기능은 전략 테스터에만 적용되며 터미널 지표는 수신된 각 틱 지표에 대해 항상 계산됩니다.
optimization_chart_mode	string	최적화 결과를 시각화하는 데 사용할 차트 유형과 두 입력 매개변수의 이름을 지정합니다. 예를 들어 "3d, InpX, InpY"는 테스트된 InpX 및 InpY 매개변수 값을 기반으로 좌표축과 함께 결과가 3D 차트에 표시됨을 의미합니다. 따라서 이 속성을 사용하면 프로그램 코드에 직접 최적화 차트와 차트 유형을 표시하는 데 사용할 모수를 지정할 수 있습니다. 가능한 옵션: • "3d, input_parameter_name1, input_parameter_name2"는 회전, 확대 및 축소할 수 있는 3D 시각화 차트를 의미합니다. 차트는 두 개의 파라미터를 사용하여 작성됩니다. • "2d, input_parameter_name1, input_parameter_name2"는 결과에 따라 각 셀이 특정 색상으로 도색되는 2D 그리드 차트를 의미합니다. 차트는 두 개의 파라미터를 사용하여 작성됩니다. • "1d, input_parameter_name1, input_parameter_name2"는 결과가 지정된 파라미터별로 정렬되는 선형 차트를 의미합니다. 각 패스가 점으로 표시됩니다. 차트는 하나의 매개변수를 기반으로 작성됩니다. • "0d, input_parameter_name1, input_parameter_name2"는 통과 결과 도착 시간별로 결과를 정렬한 정규 차트를 의미합니다. 각 패스는 차트에서 포인트로 표시됩니다. 매개 변수 표시는 필요하지 않지만 지정된 매개 변수를 다른 차트 유형으로 수동 전환하는 데 사용할 수 있습니다. 선택적으로 하나 또는 두 개의 입력 매개 변수를 지정하지 않고 차트 유형만 표시할 수 있습니다. 이 경우 터미널은 최적화 차트를 표시하는 데 필요한 파라미터를 선택합니다.

샘플 작업 설명 및 버전 번호

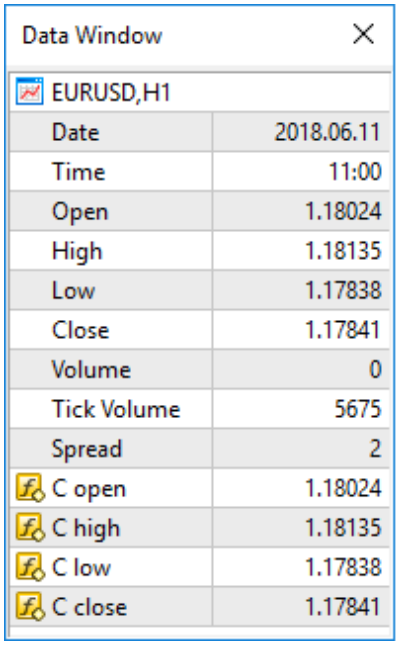
```
#property version     "3.70"      // Expert Advisor의 최신 버전
#property description "ZigZag universal with Pesavento Patterns"
#property description "현재 지표에는 알고리즘이 다른 여러 ZigZag가 포함되어 있습니다."
#property description "최고점과 최고점을 보여주는 많은 다른 지표들을 포함할 수 있습니다"
#property description "다양한 그래픽 도구를 사용하여 자동으로 구축됩니다"
```

© 2000-2025, MetaQuotes Ltd.



각 지표 버퍼에 대해 별도의 레이블 지정 예제 ("C 시가; C 고가; C 저가; C 종가")

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots   1
#property indicator_type1   DRAW_CANDLES
#property indicator_width1  3
#property indicator_label1  "C open;C high;C low;C close"
```



© 2000-2025, MetaQuotes Ltd.

파일 포함 (#include)

#include 명령문을 프로그램의 아무 곳에나 배치할 수 있지만 일반적으로 모든 포함 항목은 소스 코드의 시작 부분에 배치됩니다. 호출 형식:

```
#include <file_name>
#include "file_name"
```

예제:

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

전처리기는 *#include <file_name>* 라인들을 WinUser32.mqh 파일의 내용으로 바꿉니다. 꺽쇠 괄호는 WinUser32.mqh 파일이 표준 디렉토리(일반적으로 *terminal_installation_directory\MQL5\Include*)에서 가져올 것임을 나타냅니다. 현재 디렉터리는 검색에 포함되지 않습니다.

파일 이름이 따옴표로 묶이면 검색이 현재 디렉토리(주 원본 파일 포함)에서 이루어집니다. 표준 디렉터리는 검색에 포함되지 않습니다.

더 보기

표준 라이브러리, 임포트 함수

© 2000-2025, MetaQuotes Ltd.

임포트 함수 (#import)

기능은 컴파일된 MQL5 모듈(*.ex5 파일) 및 운영 체제 모듈(*.dll 파일)에서 가져옵니다. 모듈 이름은 *#import* 명령어에 지정되어 있습니다. 컴파일러에서 가져온 함수 호출을 올바르게 구성하고 적절한 전송 매개변수를 구성하기 위해서는 함수에 대한 전체 설명이 필요합니다. 함수 설명은 *#import "module name"* 명령문을 바로 따릅니다. 새 명령 *#import*(매개변수 없이 가능)는 가져온 함수 설명 블록을 완료합니다.

```
#import "file_name"
   func1 정의;
   func2 정의;
    ...
   funcN 정의;
#import
```

가져온 함수는 이름을 가질 수 있습니다. 이름은 같지만 모듈이 다른 함수를 동시에 가져올 수 있습니다. 가져온 함수는 기본 제공 함수의 이름과 일치하는 이름을 가질 수 있습니다. 범위 지정 연산은 호출해야 하는 함수를 정의합니다.

#import 키워드 뒤에 지정한 파일을 검색하는 순서는 임포트한 함수 호출에 설명되어 있습니다.

가져온 함수가 컴파일된 모듈 외부에 있으므로 컴파일러는 전달된 매개 변수의 유효성을 확인할 수 없습니다. 따라서 런타임 오류를 방지하려면 가져온 함수에 전달되는 파라미터의 구성 및 순서를 정확하게 설명해야 합니다. 가져온 함수(EX5 및 DLL 모듈에서 모두)에 전달되는 매개 변수는 기본값을 가질 수 있습니다.

가져온 함수의 매개 변수에는 다음을 사용할 수 없습니다:

- pointers (*);
- 동적 배열 및/또는 포인터가 포함된 개체에 대한 링크.

모든 유형의 문자열 및/또는 동적 배열을 포함하는 클래스, 문자열 배열 또는 복합 개체는 DLL에서 가져온 함수에 매개 변수로 전달할 수 없습니다.

예제:

```
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
double GetDoubleValue(double);
string GetStringValue(string);
double GetArrayItemValue(double &arr[],int,int);
bool   SetArrayItemValue(double &arr[],int,int,double);
double GetRatesItemValue(double &rates[][6],int,int,int);
#import
```

mql5 프로그램을 실행하는 동안 함수를 가져오기 위해 조기 바인딩을 사용합니다. 즉, ex5 프로그램을 사용하여 프로그램을 로드하는 동안 라이브러리가 로드됩니다.

© 2000-2025, MetaQuotes Ltd.

로드 가능한 유형의 모듈의 정규화된 이름은 사용하지 않는 것이 좋습니다 *Drive:\Directory\FileName.Ext*. MQL5 라이브러리는 *terminal_dir\MQL5\Libraries* 폴더에서 로드됩니다.

가져온 함수의 호출 버전이 32비트 및 64비트 Windows 버전에 대해 다른 경우 두 버전을 모두 가져와야 하며 _IsX64 변수를 사용하여 올바른 함수 버전을 명시적으로 호출해야 합니다.

예제:

```
#import "user32.dll"
//--- 32비트 시스템용
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
//--- 64비트 시스템용
int    MessageBoxW(ulong hWnd,string lpText,string lpCaption,uint uType);
#import
//+------------------------------------------------------------------+
//|  MessageBox_32_64_bit은 적합한 버전의 MessageBoxW()를 사용합니다   |
//+------------------------------------------------------------------+
int MessageBox_32_64_bit()
  {
   int res=-1;
   //--- 64비트 윈도우를 사용 중인 경우
   if(_IsX64)
     {
      ulong hwnd=0;
      res=MessageBoxW(hwnd,"64-bit MessageBoxW 호출 예제","MessageBoxW 64 bit",MB_OK|ME
     }
   else  // 32비트 윈도우를 사용 중인 경우
     {
      uint hwnd=0;
      res=MessageBoxW(hwnd,"32-bit MessageBoxW 호출 예제","MessageBoxW 32 bit",MB_OK|ME
     }
   return (res);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int ans=MessageBox_32_64_bit();
   PrintFormat("MessageBox_32_64_bit returned %d",ans);
  }
```

.NET 라이브러리에서의 임포트 함수

.NET 라이브러리 함수로 작업하려면 특정 함수를 정의하지 않고 DLL 자체를 임포트하면 됩니다. MetaEditor는 다음 작업을 수행할 수 있는 모든 기능을 자동으로 가져옵니다:

- 단순 구조(POD, 평범한 오랜 데이터) — 단순 데이터 유형만 포함하는 구조.
- 단순 유형 및 POD 구조 또는 해당 어레이만 사용되는 매개변수가 있는 공용 정적 함수

© 2000-2025, MetaQuotes Ltd.

라이브러리에서 함수를 호출하려면 다음과 같이 하면 됩니다:

```
#import "TestLib.dll"
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int x=41;
   TestClass::Inc(x);
   Print(x);
  }
```

TestClass의 Inc 함수의 C# 코드는 다음과 같습니다:

```
public class TestClass
  {
   public static void Inc(ref int x)
     {
      x++;
     }
  }
```

실행 결과 스크립트는 42의 값을 반환합니다.

더 보기

파일 포함

© 2000-2025, MetaQuotes Ltd.

조건부 컴파일 (#ifdef, #ifndef, #else, #endif)

프리프로세서 명령문은 컴파일하기 전에 소스 코드를 전처리하는 데 사용됩니다. 명령문은 항상 #로 시작하므로 컴파일러는 변수, 함수 등의 이름에 기호를 사용할 수 없습니다.

각 명령문은 별도의 항목으로 설명되며 줄 바꿈까지 유효합니다. 한 항목에는 여러 명령문을 사용할 수 없습니다. 지시어 입력이 너무 크면 ₩기호를 사용하여 여러 줄로 구분할 수 있습니다. 이 경우 다음 행은 명령문 입력의 연속으로 간주됩니다.

전처리기 조건부 컴파일 지침을 사용하면 특정 조건의 충족 여부에 따라 프로그램의 일부를 컴파일하거나 건너뛸 수 있습니다.

이 조건은 다음 포맷 중 하나를 사용할 수 있습니다.

```
#ifdef identifier
   // #preprocessor 지시어에 식별자가 이미 정의된 경우 여기에 있는 코드가 컴파일됩니다.
#endif
```

```
#ifndef identifier
   // 식별자가 현재 전처리기로 지시어로 #define 되지 않은 경우 여기에 있는 코드가 컴파일됩니다.
#endif
```

모든 조건부 컴파일 지시문은 #else 명령문을 포함하고 #endif로 끝나는 가능한 모든 줄의 뒤에 올 수 있습니다. 확인된 조건이 참이면 #else와 #endif 사이의 선이 무시됩니다. 확인된 조건이 충족되지 않으면 확인 및 #else 지시문(또는 전자가 없는 경우 #endif 지시문) 사이의 모든 줄이 무시됩니다.

예제:

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef TestMode
      Print("테스트 모드");
   #else
      Print("일반 모드");
   #endif
  }
```

프로그램 유형 및 컴파일 모드에 따라 표준 매크로는 다음과 같이 정의됩니다.

__MQL5__ 매크로는 *.mq5 파일을 컴파일할 때 정의되며, __MQL4__ 매크로는 *.mq4 파일을 컴파일할 때 정의됩니다.
_DEBUG 매크로는 디버그 모드에서 컴파일할 때 정의됩니다.
_RELEASE 매크로는 릴리즈 모드에서 컴파일할 때 정의됩니다.

예제:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef __MQL5__
      #ifdef _DEBUG
         Print("Hello from MQL5 compiler [DEBUG]");
      #else
        #ifdef _RELEASE
           Print("Hello from MQL5 compiler [RELEASE]");
        #endif
     #endif
   #else
      #ifdef __MQL4__
         #ifdef _DEBUG
           Print("Hello from MQL4 compiler [DEBUG]");
        #else
           #ifdef _RELEASE
              Print("Hello from MQL4 compiler [RELEASE]");
           #endif
        #endif
     #endif
   #endif
  }
```

© 2000-2025, MetaQuotes Ltd.

객체지향 프로그래밍

객체지향 프로그래밍(OOP)은 주로 데이터에 초점을 맞춘 프로그래밍이며, 데이터와 동작은 분리할 수 없이 연결되어 있습니다. 데이터와 동작이 함께 클래스를 구성하는 반면, 개체는 클래스 인스턴스입니다.

객체 지향 접근법의 구성요소는 다음과 같습니다:

- 캡슐화 및 유형 확장성
- 상속
- 다형성
- 오버로드
- 가상 함수

객체지향 프로그래밍은 연산을 동작 모델링으로 간주합니다. 모델링된 항목은 연산 추상화로 표현되는 객체입니다. 우리가 잘 알려진 게임 "테트리스"를 작성하고 싶다고 가정해봅시다. 이렇게 하려면 모서리로 결합된 네 개의 정사각형으로 구성된 랜덤 도형의 모양을 모형화하는 방법을 배워야 합니다. 또한 우리는 도형의 하강 속도를 조절하고 회전과 도형의 이동을 정의해야 합니다. 화면의 도형 이동은 우물 경계에 따라 제한되므로 이 요구사항도 모델링해야 합니다. 그 외에 채워진 큐브의 행은 파기하고 달성된 포인트를 계산해야 합니다.

따라서, 이 이해하기 쉬운 게임은 모양 모델, 우물 모델, 모양 이동 모델 등 여러 모델을 만들어야 합니다. 이 모든 모델은 추상화이며, 컴퓨터 계산으로 표현됩니다. 이러한 모델을 설명하기 위해 추상적 데이터 유형, ADT(또는 복합 데이터 유형)의 개념의 사용됩니다. 엄격히 말하면, DOM의 "모양" 모션 모델은 데이터 유형이 아니지만 "잘된" 데이터 유형의 제한을 사용하는 "모양" 데이터 유형에 대한 작업 집합입니다.

오브젝트는 class 변수입니다. 객체 지향 프로그래밍을 사용하면 ADT를 쉽게 만들고 사용할 수 있습니다. 객체 지향 프로그래밍은 상속 메커니즘을 사용합니다. 상속의 장점은 사용자가 이미 정의한 데이터 유형에서 파생 유형을 얻을 수 있다는 점입니다.

예를 들어, Tetris 쉐이프를 작성하려면 먼저 기준 클래스 쉐이프를 작성하는 것이 편리합니다. 가능한 7가지 형상 유형을 모두 나타내는 다른 클래스는 그 기준으로 파생될 수 있습니다. 쉐이프의 동작은 기준 클래스에 정의되며, 각 개별 쉐이프의 동작 구현은 파생 클래스에 정의됩니다.

OOP에서 개체는 자신의 동작을 담당합니다. ADT 개발자는 일반적으로 해당 개체에서 예상되는 동작을 설명하는 코드를 포함해야 합니다. 개체 자체가 해당 동작에 책임이 있다는 사실은 이 개체의 사용자를 위한 프로그래밍 작업을 크게 단순화합니다.

화면에 도형을 그리려면 중심이 어디에 있고 어떻게 그려야 하는지 알아야 합니다. 별도의 도형을 직접 그릴 줄 아는 경우, 프로그래머는 해당 도형을 사용할 때 "그리기" 메시지를 보내야 합니다.

MQL5 Language는 C++ 유사하며 ADT 구현을 위한 캡슐화 메커니즘도 가지고 있습니다. 한편으로 캡슐화는 특정 유형의 구현에 대한 내부 세부사항을 결합하고, 다른 한편으로는 이러한 유형의 객체에 영향을 미칠 수 있는 외부 액세스 가능 함수를 결합합니다. 이 유형을 사용하는 프로그램의 구현 세부 정보에 액세스하지 못할 수 있습니다.

OOP의 개념은 다음과 같은 일련의 관련 개념을 가지고 있습니다:

- 현실 세계의 행동 시뮬레이션
- 사용자 정의 데이터 유형
- 구현 세부 정보 숨기기

© 2000-2025, MetaQuotes Ltd.

- 상속을 통한 코드 재사용 가능
- 실행 중 함수 호출 해석

이 개념들 중 일부는상당히 모호하고, 일부는 추상적이며, 다른 것들은 일반적입니다.

© 2000-2025, MetaQuotes Ltd.

유형의 캡슐화 및 확장성

OOP는 소프트웨어를 작성하는 균형 잡힌 접근법입니다. 데이터와 동작이 함께 채워집니다. 이 캡슐화는 사용자 정의 데이터 유형을 생성하여 언어 데이터 유형을 확장하고 상호 작용합니다. 유형 확장성은 기본 유형뿐만 아니라 사용하기 쉬운 사용자 정의 데이터 유형에도 추가할 수 있는 기회입니다.

문자열과 같은 추상 데이터 유형은 잘 알려진 이상적인 동작 유형에 대한 설명입니다.

문자열 사용자는 연결 또는 인쇄와 같은 문자열 작업에 특정 동작이 있음을 알고 있습니다. 연결 및 인쇄 작업을 메서드라고 합니다.

ADT의 특정 구현에는 일부 제한이 있을 수 있습니다. 예를 들어 문자열의 길이가 제한될 수 있습니다. 이러한 제한은 모두에게 열려 있는 동작에 영향을 미칩니다. 동시에 내부 또는 프라이빗 구현 세부 정보는 사용자가 개체를 보는 방식에 직접적인 영향을 미치지 않습니다. 예를 들어 문자열은 종종 배열로 구현되지만 이 배열의 내부 기본 주소와 이름은 사용자에게 필수적이지 않습니다.

캡슐화는 사용자 정의 유형에 대한 개방형 인터페이스가 제공될 때 구현 세부 정보를 숨기는 기능입니다. MQL5에서뿐 아니라 C++에서 클래스 및 구조 정의(클래스 및 구조)는 액세스 키워드 private, protected and public과 함께 캡슐화 프로비저닝에 사용됩니다.

퍼블릭 키워드는 뒤에 있는 멤버에 대한 액세스가 제한 없이 열려 있음을 나타냅니다. 이 키워드가 없으면 클래스 멤버는 기본적으로 잠깁니다. 프라이빗 멤버는 해당 클래스의 멤버 함수만 액세스할 수 있습니다.

보호된 클래스 함수는 클래스뿐 아니라 상속자 클래스에서도 사용할 수 있습니다. 공개 클래스 기능은 클래스 선언의 범위 내의 모든 기능에 사용할 수 있습니다. 보호를 통해 클래스 구현의 일부를 숨길 수 있으므로 데이터 구조의 예기치 않은 변경을 방지할 수 있습니다. 액세스 제한 또는 데이터 숨김은 객체 지향 프로그래밍의 기능입니다.

일반적으로 클래스 함수는 protected 한정자로 보호 및 선언되며, 값의 읽기 및 쓰기는 public 액세스 한정자로 정의된 특수한 set-and-get 메서드를 사용하여 수행됩니다.

예제:

```
class CPerson
  {
protected:
   string            m_name;                    // 이름
public:
   void              SetName(string n){m_name=n;}// 이름을 설정합니다
   string            GetName(){return (m_name);} // 이름을 반환합니다
  };
```

이 접근 방식은 여러 가지 이점을 제공합니다. 첫째, 함수 이름으로 클래스 멤버의 값을 설정하거나 얻을 수 있는 기능을 이해할 수 있습니다. 둘째로, 미래에는 CPerson 클래스나 그 파생형 클래스에서 m_name 변수의 유형을 변경해야 할 것입니다.

이 경우 SetName() 및 GetName() 함수의 구현만 변경하면 되고, 사용자는 m_name의 데이터 유형이 변경되었는지도 모르기 때문에 코드 변경 없이 CPerson 클래스의 개체를 프로그램에서 사용할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

예제:

```
struct Name
  {
   string            first_name;                  // 이름
   string            last_name;                   // 성(last name)
  };

class CPerson
  {
protected:
   Name              m_name;                      // 이름
public:
   void              SetName(string n);
   string            GetName(){return(m_name.first_name+" "+m_name.last_name);}
private:
   string            GetFirstName(string full_name);
   string            GetLastName(string full_name);
  };

void CPerson::SetName(string n)
  {
   m_name.first_name=GetFirstName(n);
   m_name.last_name=GetLastName(n);
  }

string CPerson::GetFirstName(string full_name)
  {
   int pos=StringFind(full_name," ");
   if(pos>0) StringSetCharacter(full_name,pos,0);
   return(full_name);
  }

string CPerson::GetLastName(string full_name)
  {
   string ret_string;
   int pos=StringFind(full_name," ");
   if(pos>0) ret_string=StringSubstr(full_name,pos+1);
   else      ret_string=full_name;
   return(ret_string);
  }
```

더 보기

데이터 타입

© 2000-2025, MetaQuotes Ltd.

상속

OOP의 특징은 상속을 통한 코드 재사용 장려입니다. 기존 클래스에서 새 클래스가 만들어지며 이를 기준 클래스라고 합니다. 파생 클래스는 기준 클래스의 멤버를 사용하지만 이러한 멤버를 수정 및 보완할 수도 있습니다.

많은 유형이 기존 유형의 변형입니다. 그들 각각에 맞는 새로운 코드를 개발하는 것은 종종 지루할 수 있습니다. 또한 새로운 코드는 새로운 오류를 암시합니다. 파생 클래스는 기준 클래스의 설명을 상속하므로 코드를 다시 개발하고 다시 테스트할 필요가 없습니다. 상속 관계는 계층적입니다.

계층은 모든 요소의 다양성과 복잡성을 복사할 수 있는 방법입니다. 이는 객체 분류를 소개합니다. 예를 들어, 원소들의 주기율표는 가스를 가지고 있습니다. 그들은 모든 주기적인 요소에 내재된 성질을 가지고 있다.

불활성 가스는 다음으로 중요한 하위 분류를 구성합니다. 위계질서는 아르곤과 같은 불활성 가스는 기체이고, 그 다음에 가스는 시스템의 일부입니다. 그러한 계층구조는 불활성 가스의 행동을 쉽게 해석할 수 있게 한다. 우리는 그들의 원자가 양자와 전자를 포함하고 있다는 것을 알고 있습니다. 그것은 다른 모든 원소들에 해당합니다.

우리는 그들이 실온에서 가스 상태에 있다는 것을 알고 있습니다. 모든 가스처럼요. 우리는 불활성 가스 하위 등급의 가스가 다른 원소와의 일반적인 화학 반응으로 들어가지 않는다는 것을 알고 있으며, 모든 불활성 가스의 특성입니다.

기하학적 도형의 상속 예를 고려해 보십시오. 단순 도형의 전체 다양성(원, 삼각형, 직사각형, 사각형 등)을 설명하려면 파생된 모든 클래스의 조상인 기준 클래스(ADT)를 만드는 것이 가장 좋습니다.

모양을 설명하는 가장 일반적인 멤버만 포함하는 기준 클래스 CShape를 만들겠습니다. 이러한 멤버는 쉐이프의 유형 및 주 고정점 좌표 등 모든 쉐이프의 특성을 설명합니다.

예제:

```
//--- 기준 클래스 쉐이프
class CShape
  {
protected:
   int       m_type;                    // 쉐이프 유형
   int       m_xpos;                    // X - 기준점 좌표
   int       m_ypos;                    // Y - 기준점 좌표
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // constructor
   void      SetXPos(int x){m_xpos=x;} // set X
   void      SetYPos(int y){m_ypos=y;} // set Y
  };
```

다음으로 기준 클래스에서 파생된 새 클래스를 만듭니다. 여기서 각각 특정 클래스를 지정하는 필수 필드를 추가합니다. Circle 쉐이프의 경우 반지름 값을 포함하는 멤버를 추가해야 합니다. 사각형 쉐이프는 측면 값으로 설명됩니다. 따라서 기준 클래스 CShape에서 상속된 파생 클래스는 다음과 같이 선언됩니다:

```
//--- 파생 클래스 원
class CCircle : public CShape         // 콜론 뒤에 기준 클래스를 정의합니다
  {                                     // 상속할 수 있는
private:
```

© 2000-2025, MetaQuotes Ltd.

```
   int               m_radius;           // 원 반지름

public:
                     CCircle(){m_type=1;}// 구성자, type 1
  };
```

사각형 쉐이프 클래스 선언은 유사합니다:

```
//--- 파생 클래스 Square
class CSquare : public CShape         // 콜론 뒤에 기준 클래스를 정의합니다
  {                                   // 상속할 수 있는
private:
   int              m_square_side;    // 사각형 면

public:
                     CSquare(){m_type=2;} // 구성자, type 2
  };
```

객체를 생성하는 동안 기준 클래스 생성자를 먼저 호출한 다음 파생 클래스의 구성자를 호출합니다. 개체가 먼저 제거되면 파생 클래스의 소멸자가 호출된 다음 기준 클래스 소멸자가 호출됩니다.

따라서 기준 클래스에서 가장 일반적인 멤버를 선언하여 특정 클래스를 지정하는 파생 클래스에 멤버를 추가할 수 있습니다. 상속을 통해 여러 번 재사용할 수 있는 강력한 코드 라이브러리를 만들 수 있습니다.

기존 클래스에서 파생 클래스를 만드는 구문은 다음과 같습니다:

```
class class_name :
          (public | protected | private) opt  base_class_name
  {
   클래스 멤버 선언
  };
```

파생된 부류의 한 측면은 멤버 계승자(상속인)의 가시성(개방성)입니다. 퍼블릭, 보호 및 프라이빗 키워드는 파생된 클래스에 사용할 수 있는 기준 클래스의 멤버를 나타내는 데 사용됩니다. 파생 클래스 머리글의 콜론 뒤에 있는 공용 키워드는 기준 클래스 CShape의 보호된 멤버와 공용 멤버를 파생 클래스 CCircle의 보호된 멤버와 공용 멤버로 상속해야 함을 나타냅니다.

파생 클래스에 대해 기준 클래스의 프라이빗 클래스 멤버를 사용할 수 없습니다. 공용 상속은 파생 클래스(CCircle 및 CSquare)가 CShapes임을 의미하기도 합니다. 즉, 정사각형(CSquare)은 도형(CSape)이지만 도형이 반드시 정사각형일 필요는 없습니다.

파생 클래스는 기준 클래스를 수정한 것으로, 기준 클래스의 보호된 멤버와 공용 멤버를 상속합니다. 기준 클래스의 생성자 및 소멸자를 상속할 수 없습니다. 기준 클래스 멤버 외에 파생 클래스에 새로운 멤버가 추가됩니다.

파생 클래스는 기준 클래스와 다른 멤버 함수의 구현을 포함할 수 있습니다. 동일한 함수 이름의 의미가 서명마다 다를 수 있는 경우 오버로드와 공통되는 것은 없습니다.

보호된 상속에서 기준 클래스의 공용 및 보호된 멤버는 파생 클래스의 보호된 멤버가 됩니다. 프라이빗 상속에서 기준 클래스의 공용 및 보호된 멤버는 파생 클래스의 프라이빗 멤버가 됩니다.

© 2000-2025, MetaQuotes Ltd.

보호 및 비공개 상속에서 "파생형 클래스의 객체는 기준 클래스의 객체"라는 관계는 참이 아닙니다. 보호된 상속 유형 및 프라이빗 상속 유형은 드물며 각각 주의하여 사용해야 합니다.

상속 유형(공개, 보호 또는 비공개)은 **파생 클래스에서 상속 계층의 기준 클래스 멤버에 액세스**하는 방법에는 영향을 미치지 않습니다. 모든 유형의 상속에서 공개 및 보호된 액세스 지정자로 선언된 기준 클래스 멤버만 파생된 클래스에서 사용할 수 있습니다. 다음 예시에서 고려해봅시다:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| 몇가지 액세스 유형이 있는 클래스 예제                                |
//+------------------------------------------------------------------+
class CBaseClass
  {
private:              //--- 파생 클래스에서 프라이빗 멤버를 사용할 수 없습니다
   int                m_member;
protected:            //--- 보호된 메서드는 기준 클래스와 파생 클래스에서 사용할 수 있습니다
   int                Member(){return(m_member);}
public:               //--- 클래스 생성자는 모든 클래스 멤버가 사용할 수 있습니다
                      CBaseClass(){m_member=5;return;};
private:              //--- m_member에 값을 할당하는 비공개 방법
   void               Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------+
//| 오류가 있는 파생 클래스                                            |
//+------------------------------------------------------------------+
class CDerived: public CBaseClass // 기본값이므로 공용 상속의 지정을 생략할 수 있습니다
  {
public:
   void Func() // 파생 클래스에서 기준 클래스 멤버에 대한 호출을 포함하는 함수를 정의합니다
     {
      //--- 기준 클래스의 프라이빗 멤버 수정 시도
      m_member=0;         // 오류, 기준 클래스의 프라이빗 멤버를 사용할 수 없습니다
      Member(0);          // 오류, 기준 클래스의 전용 메서드를 파생 클래스에서 사용할 수 없습니
      //--- 기준 클래스의 멤버 읽기
      Print(m_member);    // 오류, 기준 클래스의 프라이빗 멤버를 사용할 수 없습니다
      Print(Member());    // 오류 없음, 보호된 메서드를 기준 클래스와 해당 파생 클래스에서 사용
     }
  };
```

위의 예에서 CBaseClass는 공용 메서드인 생성자만 있습니다. 클래스 개체를 만들 때 생성자가 자동으로 호출됩니다. 따라서 프라이빗 멤버와 보호 메소드 Member()는 외부에서 호출할 수 없습니다. 그러나 공용 상속의 경우 파생 클래스에서 기준 클래스의 Member() 메서드를 사용할 수 있습니다.

보호된 상속의 경우 공용 및 보호된 액세스 권한이 있는 기준 클래스의 모든 멤버가 보호됩니다. 이는 공공데이터 멤버와 기준 클래스의 메서드가 외부로부터 접근할 수 있고, 보호상속은 파생 클래스와 그 파생에서만 이용할 수 있다는 것을 의미합니다.

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 몇가지 액세스 유형이 있는 클래스 예제                                    |
//+------------------------------------------------------------------+
class CBaseMathClass
  {
private:              //--- 파생 클래스에서 프라이빗 멤버를 사용할 수 없습니다
   double             m_Pi;
public:               //--- m_Pi 값 가져오기 및 설정
   void               SetPI(double v){m_Pi=v;return;};
   double             GetPI(){return m_Pi;};
public:               // 클래스 생성자는 모든 멤버가 사용할 수 있습니다
                      CBaseMathClass() {SetPI(3.14);  PrintFormat("%s",__FUNCTION__);};
  };
//+------------------------------------------------------------------+
//| Derived class, in which m_Pi cannot be modified                  |
//+------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // 보호 상속
  {
private:
   double             m_radius;
public:               //--- 파생 클래스의 공용 메서드
   void               SetRadius(double r){m_radius=r; return;};
   double             GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------+
//| 스크립트 시작 함수                                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 파생 클래스를 만들 때 기준 클래스의 생성자가 자동으로 호출됩니다
   CProtectedChildClass pt;
//--- 반지름 입력
   pt.SetRadius(10);
   PrintFormat("Length=%G",pt.GetCircleLength());
//--- 아래 라인에 대한 설명을 해제하면 SetPI()가 보호되므로 컴파일 단계에서 오류가 발생합니다
// pt.SetPI(3);

//--- 이제 기준 클래스의 변수를 선언하고 Pi 상수를 10으로 설정해 보십시오
   CBaseMathClass bc;
   bc.SetPI(10);
//--- 여기 결과가 있습니다
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

이 예에서는 기준 클래스 CBaseMathClass의 SetPI() 및 GetPI() 메서드가 열려 있으며 프로그램의 모든 위치에서 호출할 수 있음을 보여 줍니다. 그러나 이와 동시에 CProtectedChildClass에서 파생된 이러한 메서드는 CProtectedChildClass 클래스 또는 해당 파생 클래스의 메서드에서만 호출할 수 있습니다.

프라이빗 상속의 경우 공인이 있는 기초계층 멤버 전원과 접근 보호가 되어 추가 상속 시 호출이 불가능해집니다.

© 2000-2025, MetaQuotes Ltd.

MQL5에는 다중 상속이 없습니다.

더 보기

구조 및 클래스

© 2000-2025, MetaQuotes Ltd.

다형성

다형성은 동일한 함수 요소를 호출할 때 서로 다른 클래스의 객체가 다양한 방식으로 반응할 수 있는 기회입니다. 기준 클래스뿐만 아니라 하위 클래스의 동작을 설명하는 범용 메커니즘을 만드는 데 도움이 됩니다.

계속해서 기준 클래스 CShape를 개발하고 쉐이프의 면적을 계산하기 위해 설계된 멤버 함수 GetArea()를 정의하겠습니다. 기준 클래스로부터 물려받아 생산되는 모든 후손 계층에서 우리는 특정 형상의 면적을 계산하는 규칙에 따라 이 기능을 재정의합니다.

사각형(클래스 CSquare)의 경우 면적을 계산하고, 원(클래스 CCircle)의 경우 면적을 반지름 등으로 표현합니다. 기준 클래스의 개체와 모든 하위 클래스의 개체를 모두 저장할 수 있는 CShape 유형의 개체를 저장할 배열을 만들 수 있습니다. 또한 어레이의 각 요소에 대해 동일한 함수를 호출할 수 있습니다.

예제:

```
//--- 기준 클래스
class CShape
  {
protected:
   int            m_type;                 // 쉐이프 유형
   int            m_xpos;                 // X - 기준점 좌표
   int            m_ypos;                 // Y - 기준점 좌표
public:
   void           CShape(){m_type=0;};    // 구성자, type=0
   int            GetType(){return(m_type);};// 쉐이프 유형 반환
virtual
   double         GetArea(){return (0); }// 쉐이프 영역 반환
  };
```

이제 파생된 모든 클래스에 0 값을 반환하는 getArea() 멤버 함수가 있습니다. 이 기능의 구현은 각 하위 항목마다 다릅니다.

```
//--- 파생 클래스 써클
class CCircle : public CShape             // 콜론 뒤에 기준 클래스를 정의합니다
  {                                       // 상속할 수 있는
private:
   double         m_radius;               // 원 반지름

public:
   void           CCircle(){m_type=1;};  // 구성자, type=1
   void           SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// 써클 영역
  };
```

클래스 Square의 경우 선언은 동일합니다:

```
//--- 파생 클래스 사각형
class CSquare : public CShape             // 콜론 뒤에 기준 클래스를 정의합니다
  {                                       // 상속할 수 있는
private:
```

© 2000-2025, MetaQuotes Ltd.

```
   double          m_square_side;        // 사각형 면

public:
   void            CSquare(){m_type=2;}; // constructor, type=1
   void            SetSide(double s){m_square_side=s;};
   virtual double  GetArea(){return (m_square_side*m_square_side);}// 사각형 면적
  };
```

사각형과 원의 면적을 계산하려면 m_radius와 m_square_side의 해당 값이 필요하므로 해당 클래스의 선언에 SetRadius()와 SetSide() 함수를 추가했습니다.

하나의 기준 타입 CShape에서 파생된 서로 다른 타입의 객체(CCircle과 CSquare)가 우리 프로그램에서 사용된다고 가정합니다. 다형성을 사용하면 기준 CShape 클래스의 개체 배열을 만들 수 있지만 이 배열을 선언할 때 이러한 개체는 아직 알 수 없으며 유형이 정의되지 않습니다.

배열의 각 요소에 포함될 개체 유형에 대한 결정은 프로그램 실행 중에 직접 이루어집니다. 여기에는 적절한 클래스의 개체의 동적 생성이 포함되므로 개체 대신 객체 포인터를 사용해야 합니다.

new 연산자는 객체의 동적 작성에 사용됩니다. 이러한 각 객체는 delete 연산자를 사용하여 개별적으로 명시적으로 삭제해야 합니다. 따라서 다음 스크립트 예제와 같이 CShape 유형의 포인터 배열을 선언하고 각 요소 (**new Class_Name**)에 대해 적절한 유형의 개체를 만듭니다:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 기준 유형의 개체 포인터 배열을 선언
   CShape *shapes[5];   // CShape 개체에 대한 포인터 배열

//--- 여기서 배열을 파생 객체로 채움
//--- CCircle 유형의 개체에 대한 포인터를 선언
   CCircle *circle=new CCircle();
//--- 원형 포인터에서 객체 특성 설정
   circle.SetRadius(2.5);
//--- 포인터 값을 shapes[0]에 배치
   shapes[0]=circle;

//--- 다른 CCircle 객체를 만들고 포인터를 shapes[1]에 기록
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);

//--- 여기서는 shapes[2]에 대한 값을 설정하기 위해 의도적으로 "잊음"
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- 사용되지 않는 요소에 대해 NULL 설정
   shapes[2]=NULL;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- CSquare 개체를 만들고 해당 포인터를 shapes[3]에 기록
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- CSquare 개체를 만들고 해당 포인터를 shapes[4]에 기록
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- 포인터가 여러 개 있는데 크기를 파악해야 합니다
   int total=ArraySize(shapes);
//--- 배열의 모든 포인터를 통해 루프를 전달
   for(int i=0; i<5;i++)
     {
      //--- 지정된 인덱스의 포인터가 유효한 경우
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- 쉐이프의 유형 및 사각형 기록
         PrintFormat("%d 유형의 객체가 %G 정사각형을 갖습니다.",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- 포인터가 POINTER_INVALID 유형이 있는 경우
      else
        {
         //--- 오류 통지
         PrintFormat("객체 shapes[%d]이(가) 초기화되지 않았습니다! 포인트는 %s 입니다",
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- 생성된 모든 동적 개체를 삭제해야 합니다
   for(int i=0;i<total;i++)
     {
      //--- POINTER_DYNAMIC 유형의 포인터가 있는 개체만 삭제할 수 있습니다.
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- 삭제 통지
         PrintFormat("shapes[%d] 삭제",i);
         //--- 포인터로 객체 삭제
         delete shapes[i];
        }
     }
  }
```

삭제 연산자를 사용하여 객체를 삭제할 때, 포인터 유형을 선택해야 합니다. POINTER_DYNAMIC 포인터가 있는 개체만 삭제를 사용하여 삭제할 수 있습니다. 다른 유형의 포인터의 경우 오류가 반환됩니다.

© 2000-2025, MetaQuotes Ltd.

그러나 다형성은 상속 중 함수의 재정의 외에도 클래스 내에서 서로 다른 매개변수 집합을 가진 동일한 함수의 구현도 포함합니다. 즉, 클래스에는 이름은 같지만 유형 및/또는 매개변수 집합이 다른 여러 함수가 있을 수 있습니다. 이 경우 다형성은 함수 오버로드를 통해 구현됩니다.

더 보기

표준 라이브러리

© 2000-2025, MetaQuotes Ltd.

오버로드

한 클래스 내에서 동일한 이름을 사용하지만 매개 변수의 수는 서로 다른 두 개 이상의 메서드를 정의할 수 있습니다. 이 경우 메서드를 오버로드라고 하며 이러한 프로세스를 메서드 오버로드라고 합니다.

메서드 오버로딩은 다형성 실현의 한 방법입니다. 메서드의 오버로드는 함수 오버로드와 동일한 규칙에 따라 수행됩니다.

호출된 함수가 정확히 일치하지 않는 경우 컴파일러는 다음 세 가지 수준에서 적합한 함수를 순차적으로 검색합니다:

1. 클래스 메서드 내에서 검색합니다.
2. 기준 클래스 메서드 내에서 검색하며 가장 가까운 상위 항목부터 가장 처음 항목까지 일관되게 검색합니다.
3. 여러 기능 중에서 검색합니다.

모든 수준에서 정확한 대응 관계가 없지만 여러 수준에서 적합한 함수가 발견되면 최소 수준에서 발견된 함수가 사용됩니다. 한 수준 내에 적합한 기능이 하나만 있을 수 있습니다.

더 보기

함수 오버로드

© 2000-2025, MetaQuotes Ltd.

가상 함수

가상 키워드는 함수 지정자로, 기본 및 파생 클래스의 함수 중 적절한 함수 멤버를 런타임에 동적으로 선택할 수 있는 메커니즘을 제공합니다. 구조에는 가상 함수를 사용할 수 없습니다. 함수 멤버에 대해서만 선언을 변경할 수 있습니다.

가상 함수는 일반 함수와 마찬가지로 실행가능한 바디를 가져야 합니다. 호출 시 시맨틱은 다른 함수와 동일합니다.

파생 클래스에서 가상 함수를 재정의할 수 있습니다. 가상 함수에 대해 호출해야 하는 함수정의를 동적으로 선택합니다(실시간). 기준 클래스에 가상 함수가 포함되어 있고 파생 클래스에 이 함수의 자체 버전이 있는 경우가 대표적입니다.

기준 클래스에 대한 포인터는 기준 클래스 개체 또는 파생 클래스의 개체를 나타낼 수 있습니다. 호출할 멤버 함수의 선택은 런타임에 수행되며 포인터 유형이 아닌 개체 유형에 따라 달라집니다. 파생 형식의 멤버가 없는 경우 기본적으로 기준 클래스의 가상 함수가 사용됩니다.

소멸자는 가상 키워드로 선언되었는지 여부에 관계없이 항상 가상입니다.

MT5_Tetris.mq5의 예에서는 가상 함수의 사용을 살펴보겠습니다. 포함된 파일 MT5_TetrisShape.mqh에 그리기 기능을 가진 기본 클래스 CTetrisShape가 정의되어 있습니다.

```
//+------------------------------------------------------------------+
class CTetrisShape
  {
protected:
   int               m_type;
   int               m_xpos;
   int               m_ypos;
   int               m_xsize;
   int               m_ysize;
   int               m_prev_turn;
   int               m_turn;
   int               m_right_border;
public:
   void              CTetrisShape();
   void              SetRightBorder(int border) { m_right_border=border; }
   void              SetYPos(int ypos)          { m_ypos=ypos;           }
   void              SetXPos(int xpos)          { m_xpos=xpos;           }
   int               GetYPos()                  { return(m_ypos);        }
   int               GetXPos()                  { return(m_xpos);        }
   int               GetYSize()                 { return(m_ysize);       }
   int               GetXSize()                 { return(m_xsize);       }
   int               GetType()                  { return(m_type);        }
   void              Left()                     { m_xpos-=SHAPE_SIZE;    }
   void              Right()                    { m_xpos+=SHAPE_SIZE;    }
   void              Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3) r
   virtual void      Draw()                     { return;                }
   virtual bool      CheckDown(int& pad_array[]);
   virtual bool      CheckLeft(int& side_row[]);
   virtual bool      CheckRight(int& side_row[]);
```

© 2000-2025, MetaQuotes Ltd.

```
  };
```

또한 각 파생 클래스에 대해 이 기능은 하위 세분류의 특성에 따라 구현됩니다. 예를 들어 첫 번째 형상 CTetrisShape1에는 Draw() 함수가 구현되어 있습니다.

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- 쉐이프 그리기
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- 수평적
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
           }
        }
      else
        {
         //--- 수직적
         for(i=0; i<4; i++)
           {
            name=SHAPE_NAME+(string)i;
            ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
            ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
           }
        }
    }
  }
```

사각형 쉐이프는 클래스 CTetrisShape6에 의해 설명되며 자체적인 Draw() 메서드가 있습니다:

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- 쉐이프 그리기
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      for(i=0; i<2; i++)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
        }
      for(i=2; i<4; i++)
        {
         name=SHAPE_NAME+(string)i;
         ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
         ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
        }
    }
  };
```

생성된 개체가 속한 클래스에 따라 이 클래스 또는 저 파생 클래스의 가상 함수를 호출합니다.

```
void CTetrisField::NewShape()
  {
//--- 7개의 가능한 도형 중 하나를 임의로 만듭니다
   int nshape=rand()%7;
   switch(nshape)
     {
      case 0: m_shape=new CTetrisShape1; break;
      case 1: m_shape=new CTetrisShape2; break;
      case 2: m_shape=new CTetrisShape3; break;
      case 3: m_shape=new CTetrisShape4; break;
      case 4: m_shape=new CTetrisShape5; break;
      case 5: m_shape=new CTetrisShape6; break;
      case 6: m_shape=new CTetrisShape7; break;
     }
//--- 그리기
   m_shape.Draw();
//---
  }
```

한정자 '오버라이드'

'override' 한정자는 선언된 함수가 부모 클래스의 메서드를 재정의해야 함을 의미합니다. 이 방법을 사용하면 오류 재정의가 발생하지 않도록 할 수 있습니다. 예를 들어 메서드 서명이 실수로 수정되는 것을 방지할 수 있습니다. 기본 클래스에 'func' 메서드가 정의되어 있다고 가정합니다. 메서드에서는 int 변수를 인수로 받아들입니다:

```
class CFoo
  {
   void virtual func(int x) const { }
  };
```

그런 다음 자식 클래스에서 메서드를 재정의합니다:

```
class CBar : public CFoo
  {
   void func(short x) { }
  };
```

© 2000-2025, MetaQuotes Ltd.

그러나 인수 유형이 int에서 short로 잘못 변경되었습니다. 사실 이것은 메서드를 재정의하는 것이 아니라 메서드 오버로드입니다. 오버로드된 함수 알고리즘 정의에 따라 작동하는 컴파일러는 특정 상황에서 재정의된 메서드 대신 기본 클래스에 정의된 메서드를 선택할 수 있습니다.

이러한 오류를 방지하려면 재정의할 메서드에 'override' 한정자를 명시적으로 추가해야 합니다.

```
class CBar : public CFoo
  {
   void func(short x) override { }
  };
```

재정의하는 동안 메서드 서명이 변경되면 컴파일러는 상위 클래스에서 동일한 서명을 가진 메서드를 찾을 수 없으며 컴파일 오류를 반환합니다:

```
'CBar::func' 메서드가 'override' 지정자로 선언되었지만 기준 클래스 메서드는 재정의하지 않습니다
```

한정자 'final'

'final' 한정자는 이와 반대입니다. 자식 클래스에서 메서드를 재정의할 수 없습니다. 메서드 구현이 충분하고 완전히 완료된 경우 나중에 수정되지 않도록 'final' 한정자를 사용하여 이 메서드를 선언하십시오.

```
class CFoo
  {
   void virtual func(int x) final { }
  };

class CBar : public CFoo
  {
   void func(int) { }
  };
```

위의 예와 같이 메서드를 'final' 한정자로 재정의하려고 하면 컴파일러에서 오류를 반환합니다:

```
'CFoo::func' 메서드를 'CBar::func'로 재정의할 수 없습니다.
'CFoo::func' 선언을 참조하십시오.
```

더 보기

표준 라이브러리

© 2000-2025, MetaQuotes Ltd.

클래스/구조의 정적 멤버

스태틱 멤버

스토리지 클래스 한정자 static을 사용하여 클래스의 멤버를 선언할 수 있습니다. 이러한 데이터 멤버는 이 클래스의 모든 인스턴스가 공유하며 한 곳에 저장됩니다. 각 클래스 개체 변수에 대해 정적이 아닌 데이터 멤버가 생성됩니다.

클래스의 정적 멤버를 선언할 수 없는 경우 이러한 데이터를 프로그램의 글로벌 수준에 선언해야 합니다. 데이터와 해당 클래스 간의 관계를 끊을 수 있으며, OOP의 기본 패러다임인 데이터와 클래스 내 데이터 처리 방법과 일치하지 않습니다. 정적 멤버를 사용하면 특정 인스턴스에 특정되지 않은 클래스 데이터가 클래스 범위에 존재할 수 있습니다.

정적 클래스 멤버는 특정 인스턴스에 종속되지 않으므로 이에 대한 참조는 다음과 같습니다:

```
class_name::variable
```

여기서 *class_name*는 클래스의 이름이고 *variable*은 클래스 멤버의 이름입니다.

보시다시피 클래스의 정적 멤버에 액세스하기 위해, context resolution 연산자 ::가 사용됩니다. 클래스 메서드 내에서 정적 멤버에 액세스할 때 컨텍스트 연산자는 선택 사항입니다.

클래스의 정적 멤버를 원하는 값으로 명시적으로 초기화해야 합니다. 이를 위해 전역 범위에서 선언 및 초기화해야 합니다. 정적 멤버 초기화 순서는 전역 범위에서 해당 선언의 순서에 해당합니다.

예를 들어, 텍스트 구문 분석 시 사용되는 클래스 *CParser*가 있으며, 처리된 총 단어와 문자 수를 세어야 합니다. 필요한 클래스 멤버를 정적 클래스 멤버로만 선언하고 글로벌 레벨에서 초기화하면 됩니다. 그러면 클래스의 모든 인스턴스가 단어와 문자의 공통 카운터를 사용합니다.

```
//+------------------------------------------------------------------+
//| 클래스 "Text analyzer"                                           |
//+------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
   //--- 구성자 및 소멸자
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- 글로벌 수준에서 구문 분석기 클래스의 정적 멤버 초기화
int CParser::s_words=0;
int CParser::s_symbols=0;
```

정적 클래스 멤버를 *const* 키워드로 선언할 수 있습니다. 이러한 정적 상수는 *const* 키워드로 글로벌 수준에서 초기화해야 합니다:

```
//+------------------------------------------------------------------+
//| 처리된 데이터를 저장하기 위한 클래스 "Stack"                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
class CStack
  {
public:
                     CStack(void);
                    ~CStack(void){};
...
private:
   static const int  s_max_length; // 최대 스택 용량
  };

//--- CStack 클래스의 정적 상수 초기화
const int CStack::s_max_length=1000;
```

this 포인터

키워드 this는 메서드가 실행되는 클래스의 특정 인스턴스에 암시적으로 선언된 pointer를 나타냅니다. 클래스의 정적이 아닌 메서드에서만 사용할 수 있습니다. 포인터 this는 모든 클래스의 암시적 비정적 멤버입니다.

정적 함수에서는 클래스의 정적 멤버/메소드만 액세스할 수 있습니다.

정적 메서드

MQL5에서 static 유형의 멤버 함수를 사용할 수 있습니다. *static* 한정자는 클래스 내부의 선언에서 함수의 반환 유형 앞에 와야 합니다.

```
class CStack
  {
public:
   //--- 구성자 및 소멸자
                     CStack(void){};
                    ~CStack(void){};
   //--- 최대 스택 용량
   static int        Capacity();
private:
   int               m_length;     // 스택의 요소 수
   static const int  s_max_length; // 최대 스택 용량
  };
//+------------------------------------------------------------------+
//| 스택에 저장할 최대 요소 수를 반환합니다     |
//+------------------------------------------------------------------+
int CStack::Capacity(void)
  {
   return(s_max_length);
  }
//--- CStack 클래스의 정적 상수 초기화
const int CStack::s_max_length=1000;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- CStack 유형 변수 선언
   CStack stack;
//--- 객체의 정적인 방법을 호출
   Print("CStack.s_max_length=",stack.Capacity());
//--- 메서드가 정적이며 개체의 존재가 필요하지 않으므로 다음 방법으로도 호출할 수 있습니다
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

const 한정자가 있는 메서드를 상수라고 하며 클래스의 암시적 멤버를 수정할 수 없습니다. 클래스 및 상수 매개변수의 상수 함수 선언을 *const-correctness* 컨트롤이라고 합니다. 이 컨트롤을 통해 컴파일러는 개체 값의 일관성을 보장하고 컴파일 중에 잘못된 것이 있으면 오류를 반환할 수 있습니다.

const 한정자는 클래스 선언 내의 인수 목록 뒤에 배치됩니다. 클래스 외부의 정의에는 *const* 한정자도 포함되어야 합니다:

```
//+------------------------------------------------------------------+
//| 클래스 "Rectangle"                                               |
//+------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_width;       // 너비
   double            m_height;      // 높이
public:
   //--- 생성자 및 소멸자
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double h):m_width(w),m_height(h)
                    ~CRectangle(void){};
   //--- 면적 계산
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------+
//| "Rectangle" 객체의 영역을 반환합니다                             |
//+------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
  }
//+------------------------------------------------------------------+
//| 두 변수의 곱을 반환합니다                                        |
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
  {
   return(w*h);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 변이 5와 6인 직사각형을 생성합니다
   CRectangle rect(5,6);
//--- 상수 방법을 사용하여 사각형 영역 찾기
   PrintFormat("rect.Square()=%.2f",rect.Square());
//--- 클래스 CRectangle의 정적 메서드를 사용하여 숫자의 곱을 구합니다.
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

항상성 컨트롤을 사용하는 데 유리한 추가 인수는 이 경우 컴파일러에서 특별한 최적화를 생성한다는 점입니다(예: 상수 개체를 읽기 전용 메모리에 배치).

const 한정자로 정적 함수를 확인할 수 없습니다. 이 한정자는 이 함수를 호출할 때 인스턴스 멤버의 항상성을 보장하기 때문입니다. 그러나 위에서 언급한 것처럼 정적 함수는 정적 클래스 멤버가 아닌 클래스에 액세스할 수 없습니다.

더 보기

정적 변수, 변수, 레퍼런스. 한정자 & 및 키워드 this

© 2000-2025, MetaQuotes Ltd.

함수 템플릿

오버로드 함수는 일반적으로 다양한 데이터 유형에 대해 유사한 작업을 수행하는 데 사용됩니다. ArraySize()는 MQL5에서 이러한 함수의 간단한 예입니다. 이는 모든 유형의 배열 크기를 반환합니다. 실제로 이 시스템 기능은 오버로드 상태이며 이러한 오버로드의 전체 구현은 MQL5 애플리케이션 개발자에게 숨겨져 있습니다:

```
int  ArraySize(
   void&  array[]      // 체크된 배열
   );
```

즉, MQL5 언어 컴파일러는 이 함수의 각 호출에 필요한 구현을 삽입합니다. 예를 들어, 정수형 배열에 대해 다음과 같이 수행할 수 있습니다:

```
int  ArraySize(
   int&  array[]      // int 형식 요소가 있는 배열
   );
```

ArraySize() 함수는 과거 데이터 형식의 따옴표로 작업하기 위해 MqlRates 유형 배열에 대해 다음과 같은 방법으로 표시할 수 있습니다:

```
int  ArraySize(
   MqlRates&  array[] // MqlRates 유형 값으로 채워진 배열
   );
```

따라서 동일한 기능을 사용하여 다른 유형의 작업을 수행하는 것이 매우 편리합니다. 그러나 모든 예비 작업을 수행해야 합니다. 필요한 기능은 올바르게 작동해야 하는 모든 데이터 유형에 대해 오버로드되어야 합니다.

편리한 해결책이 있습니다. 각 데이터 유형에 대해 유사한 작업을 실행해야 하는 경우 함수 템플릿을 사용할 수 있습니다. 이 경우 프로그래머는 하나의 기능 템플릿 설명만 작성하면 됩니다. 그러한 방법으로 템플릿을 설명하는, 우리는 어떤 일정한 데이터 기능과 협력해야 합니다. 대신에 단지 일부 형식 매개 변수가 지정되어야 합니다. 컴파일러는 함수를 호출할 때 사용되는 인수 유형에 따라 각 유형의 적절한 처리를 위한 다양한 함수를 자동으로 생성합니다.

함수 템플릿 정의는 template 키워드 다음에 대괄호 안의 형식 매개 변수 목록으로 시작합니다. 각 형식 파라미터 앞에는 typename 키워드가 표시됩니다. 형식 매개 변수 유형은 기본 제공 유형 또는 사용자 정의 유형입니다. 다음과 같이 사용됩니다:

- 함수 인수 유형을 지정;
- 함수의 반환 값 유형을 지정,
- 함수 정의 내에서 변수를 선언

템플릿 매개 변수 수는 8개를 초과할 수 없습니다. 템플릿 정의의 각 형식 매개 변수는 함수 매개 변수 목록에 한 번 이상 나타나야 합니다. 형식 매개 변수의 각 이름은 고유해야 합니다.

다음은 모든 숫자 유형(정수 및 실수)의 배열에서 가장 높은 값을 검색하는 함수 템플릿의 예입니다:

```
template<typename T>
T ArrayMax(T &arr[])
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   uint size=ArraySize(arr);
   if(size==0) return(0);

   T max=arr[0];
   for(uint n=1;n<size;n++)
      if(max<arr[n]) max=arr[n];
//---
   return(max);
  }
```

이 템플릿은 전달된 배열에서 가장 높은 값을 찾고 그 결과로 이 값을 반환하는 함수를 정의합니다. MQL5에 내장된 ArrayMaximum() 함수는 값 자체를 찾는 데 사용할 수 있는 가장 높은 값 인덱스만 반환합니다. 예:

```
//--- 배열 생성
   double array[];
   int size=50;
   ArrayResize(array,size);
//---  랜덤 값으로 채움
   for(int i=0;i<size;i++)
     {
      array[i]=MathRand();
     }

//--- 배열에서 가장 높은 값의 위치 찾기
   int max_position=ArrayMaximum(array);
//--- 이제 어레이에서 가장 높은 값 자체를 가져옵니다
   double max=array[max_position];
//--- 찾은 값 표시
   Print("Max value = ",max);
```

따라서 어레이에서 가장 높은 값을 얻기 위해 두 단계를 수행했습니다. ArrayMax() 함수 템플릿을 사용하면 적절한 유형의 배열을 이 함수에 전달하는 것만으로도 필요한 유형의 결과를 얻을 수 있습니다. 마지막 두 줄 대신을 뜻합니다

```
//--- 배열에서 가장 높은 값의 위치 찾기
   int max_position=ArrayMaximum(array);
//--- 이제 어레이에서 가장 높은 값 자체를 가져옵니다
   double max=array[max_position];
```

이제 반환된 결과의 유형이 함수에 전달된 배열과 동일한 한 줄만 사용할 수 있습니다:

```
//--- 가장 높은 값을 발견
   double max=ArrayMax(array);
```

이 경우 ArrayMax() 함수에 의해 반환되는 결과 유형이 배열 유형과 자동으로 일치합니다.

typename 키워드를 사용하여 인수 유형을 문자열로 가져와 다양한 데이터 유형으로 작업하는 범용 메서드를 만들 수 있습니다. 데이터 형식을 문자열로 반환하는 함수의 구체적인 예를 살펴보겠습니다:

```
#include <Trade\Trade.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CTrade trade;
   double d_value=M_PI;
   int i_value=INT_MAX;
   Print("d_value: type=",GetTypeName(d_value), ",   value=", d_value);
   Print("i_value: type=",GetTypeName(i_value), ",   value=", i_value);
   Print("trade: type=",GetTypeName(trade));
//---
  }
//+------------------------------------------------------------------+
//| 유형이 라인으로 반환됩니다                                       |
//+------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- 유형을 행으로 반환
   return(typename(T));
//---
  }
```

함수 템플릿은 다음과 같은 클래스 메서드에도 사용할 수 있습니다:

```
class CFile
  {
   ...
public:
   ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
  {
   ...
   return(FileWriteStruct(m_handle,data));
  }
```

함수 테플릿은 export, virtual 및 #import 키워드로 선언하면 안 됩니다.

템플릿 함수 오버로드

© 2000-2025, MetaQuotes Ltd.

때로는 템플릿 함수 오버로드가 필요할 수 있습니다. 예를 들어, typecasting을 사용하여 두 번째 파라미터의 값을 첫 번째 파라미터에 쓰는 템플릿 함수가 있습니다. MQL5에서는 string - bool 타입캐스팅을 사용할 수 없습니다. 템플릿 함수의 오버로드를 생성해 보겠습니다. 예:

```
//+------------------------------------------------------------------+
//| 템플릿 함수                                                      |
//+------------------------------------------------------------------+
template<typename T1,typename T2>
string Assign(T1 &var1,T2 var2)
  {
   var1=(T1)var2;
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| bool+string용 특별 오버로드                                      |
//+------------------------------------------------------------------+
string Assign(bool &var1,string var2)
  {
   var1=(StringCompare(var2,"true",false) || StringToInteger(var2)!=0);
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int i;
   bool b;
   Print(Assign(i,"test"));
   Print(Assign(b,"test"));
  }
```

코드 실행 결과 int+string 페어에 Assign() 템플릿 함수가 사용되었고 오버로드된 버전은 두 번째 호출 중에 이미 bool+string 페어에 사용되었음을 알 수 있습니다.

```
string Assign<int,string>(int&,string)
string Assign(bool&,string)
```

더 보기

오버로드

© 2000-2025, MetaQuotes Ltd.

템플릿 이점

함수 템플릿 은 배열에서 최대 요소 검색과 같이 다양한 데이터 유형에 대해 유사한 작업을 수행해야 할 때 사용됩니다. 템플릿을 적용하는 가장 큰 장점은 각 유형에 대해 별도의 오버로드코드를 지정할 필요가 없다는 것입니다. 각 유형의 오버로드를 여러 개 선언하는 대신

```
double ArrayMax(double array[])
  {
   ...
  }
int ArrayMax(int array[])
  {
   ...
  }
uint ArrayMax(uint array[])
  {
   ...
  }
long ArrayMax(long array[])
  {
   ...
  }
datetime ArrayMax(datetime array[])
  {
   ...
  }
```

템플릿 함수는 하나만 작성하면 됩니다

```
template<typename T>
T ArrayMax(T array[])
  {
   if(ArraySize()==0)
      return(0);
   uint max_index=ArrayMaximum(array);
   return(array[max_index]);
  }
```

코드에 사용하기 위해:

```
double high[];
datetime time[];
....
double max_high=ArrayMax(high);
datetime lasttime=ArrayMax(time);
```

여기서 사용된 데이터 유형을 지정하는 *T* 형식 매개변수는 컴파일하는 동안 실제로 적용된 형식으로 대체됩니다. 즉, 컴파일러는 각 유형 (double, datetime, 등)에 대해 별도의 함수를 자동으로 생성합니다. 또한 MQL5를 사용하면 접근 방식의 모든 장점을 사용하여 클래스 템플릿을 개발할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

클래스 템플릿

클래스 템플릿은 template 키워드를 사용하여 선언되며, 형식 매개 변수 목록을 열거하는 꺾쇠 괄호 <> 를 사용하여 typename 키워드와 함께 선언됩니다. 이 항목은 컴파일러에게 클래스를 구현할 때 실제 변수 유형을 정의하는 T 형식 매개 변수를 가진 일반 클래스를 처리함을 알려줍니다. 예를 들어, T 유형 요소가 있는 배열을 저장하기 위한 벡터 클래스를 생성하겠습니다:

```
#define TOSTR(x) #x+" "   // 객체명 표시 매크로
//+------------------------------------------------------------------+
//| T형 요소 저장을 위한 벡터 클래스                                 |
//+------------------------------------------------------------------+
template <typename T>
class TArray
  {
protected:
   T                 m_array[];
public:
   //--- 생성자는 기본적으로 10개 요소의 배열을 작성합니다
   void TArray(void){ArrayResize(m_array,10);}
   //--- 지정된 배열 크기를 사용하여 벡터 생성자
   void TArray(int size){ArrayResize(m_array,size);}
   //--- TARray 유형 개체에 저장된 데이터 유형과 양을 반환합니다
   string Type(void){return(typename(m_array[0])+":"+(string)ArraySize(m_array));};
  };
```

다음으로, 다양한 유형으로 작업할 수 있도록 프로그램에서 세 개의 *TArray* 개체를 만들기 위해 다른 방법을 적용하겠습니다

```
void OnStart()
  {
   TArray<double> double_array;   // 벡터의 기본 크기는 10입니다
   TArray<int> int_array(15);     // 벡터의 크기는 15입니다
   TArray<string> *string_array;  // TArray<string> 벡터에 대한 포인터
//--- 동적 객체 생성
   string_array=new TArray<string>(20);
//--- 저널에 객체 이름, 데이터 유형 및 벡터 크기 표시
   PrintFormat("%s (%s)",TOSTR(double_array),double_array.Type());
   PrintFormat("%s (%s)",TOSTR(int_array),int_array.Type());
   PrintFormat("%s (%s)",TOSTR(string_array),string_array.Type());
//--- 프로그램을 완료하기 전에 동적 객체 제거
   delete(string_array);
  }
```

스크립트 실행 결과:

```
  double_array  (double:10)
  int_array  (int:15)
  string_array  (string:20)
```

이제 데이터 유형이 다른 벡터 3개가 있습니다: double, int, string.

© 2000-2025, MetaQuotes Ltd.

클래스 템플릿은 모든 유형의 다른 개체를 캡슐화하기 위해 설계된 개체인 컨테이너 개발에 적합합니다. 컨테이너 개체는 이미 하나의 특정 유형의 개체를 포함하는 컬렉션입니다. 일반적으로 저장된 데이터로 작업하는 것은 컨테이너에 즉시 내장됩니다.

예를 들어, 배열 외부에 있는 요소에 대한 액세스를 허용하지 않는 클래스 템플릿을 작성하여 "범위를 벗어남" 심각 오류를 방지할 수 있습니다.

```
//+------------------------------------------------------------------+
//| 배열 요소에 대한 자유 액세스를 위한 클래스                          |
//+------------------------------------------------------------------+
template<typename T>
class TSafeArray
  {
protected:
   T                 m_array[];
public:
   //--- 기본 구성자
   void              TSafeArray(void){}
   //--- 지정된 크기의 배열을 만들기 위한 생성자
   void              TSafeArray(int size){ArrayResize(m_array,size);}
   //--- 배열 크기
   int               Size(void){return(ArraySize(m_array));}
   //--- 배열 크기 변경
   int               Resize(int size,int reserve){return(ArrayResize(m_array,size,rese
   //--- 배열 릴리즈
   void              Erase(void){ZeroMemory(m_array);}
   //--- 인덱스로 배열 요소에 액세스하기 위한 연산자
   T                 operator[](int index);
   //--- 배열에서 모든 요소를 한 번에 수신하기 위한 대입 연산자
   void              operator=(const T  &array[]); // T 유형 배열
  };
//+------------------------------------------------------------------+
//| 인덱스별 요소 수신                                                |
//+------------------------------------------------------------------+
template<typename T>
T TSafeArray::operator[](int index)
  {
   static T invalid_value;
//---
   int max=ArraySize(m_array)-1;
   if(index<0 || index>=ArraySize(m_array))
     {
      PrintFormat("%s 인덱스 %d 은(는) 다음 범위에 없습니다: (0-%d)!",__FUNCTION__,index,r
      return(invalid_value);
     }
//---
   return(m_array[index]);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 배열에 할당                                                      |
//+------------------------------------------------------------------+
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   int size=ArraySize(array);
   ArrayResize(m_array,size);
//--- T 유형이 복사 연산자를 지원해야 합니다
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int copied,size=15;
   MqlRates rates[];
//--- 값 배열 복제
   if((copied=CopyRates(_Symbol,_Period,0,size,rates))!=size)
     {
      PrintFormat("CopyRates(%s,%s,0,%d) 가 %d 에러코드를 반환했습니다",
      _Symbol,EnumToString(_Period),size,GetLastError());
      return;
     }
//--- 컨테이너를 만들고 MqlRates 값 배열을 컨테이너에 삽입합니다
   TSafeArray<MqlRates> safe_rates;
   safe_rates=rates;
   //--- 배열 내에서 색인화
   int index=3;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
   //--- 배열 외부 인덱스
   index=size;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
  }
```

클래스 선언 외부의 메서드를 설명할 때도 템플릿 선언을 사용해야 합니다:

```
template<typename T>
T TSafeArray::operator[](int index)
  {
   ...
  }
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   ...
  }
```

© 2000-2025, MetaQuotes Ltd.

클래스 및 함수 템플릿을 사용하여 "키 – 값" 쌍을 저장하기 위한 맵 컬렉션과 같이 쉼표로 구분된 여러 형식 매개변수를 정의할 수 있습니다:

```
template<typename Key, template Value>
class TMap
  {
   ...
  }
```

더 보기

함수 템플릿, 오버로드

© 2000-2025, MetaQuotes Ltd.

추상 클래스 및 순수 가상 함수

추상 클래스는 일반 요소를 작성하는 데 사용되며, 일반 요소는 보다 구체적인 파생 클래스를 작성하는 데 사용됩니다. 추상 클래스는 일부 다른 클래스의 기준 클래스로만 사용할 수 있으므로 추상 클래스 유형의 개체를 만들 수 없습니다.

하나 이상의 순수 가상 함수를 포함하는 클래스는 추상적입니다. 따라서 추상 클래스에서 파생된 클래스는 모든 순수 가상 함수를 구현해야 하며, 그렇지 않으면 추상 클래스도 됩니다.

가상 함수는 순수 지정자 구문을 사용하여 "순수"로 선언됩니다. 공통 기능을 제공하기 위해 만들어진 CAnimal 클래스의 예를 생각해 보십시오. CAnimal 유형의 개체는 실용적으로 사용하기에는 너무 일반적입니다. 따라서, CAnimal은 추상적 클래스의 좋은 예입니다.

```
class CAnimal
  {
public:
                     CAnimal();      // 구성자
   virtual void      Sound() = 0;    // 순수 가상 함수
private:
   double            m_legs_count;   // 동물의 다리 수
  };
```

여기서 Sound()는 순수 가상 함수 PURE(**=0**)의 지정자로 선언되므로 순수 가상 함수입니다.

순수 가상 함수는 PURE 지정자가 (=NULL) 또는 (=0)으로 설정된 가상 함수입니다. 추상 클래스 선언 및 사용 예제:

```
class CAnimal
  {
public:
   virtual void      Sound()=NULL;   // PURE 메서드, 파생 클래스에서 재정의되어야 합니다.
  };
//--- 추상 클래스에서 파생됨
class CCat : public CAnimal
 {
public:
  virtual void       Sound() { Print("Myau"); } // PURE가 재정의되었습니다. CCat은 추상이
 };

//--- 잘못된 사용 예
new CAnimal;          // 'CAnimal' 오류 - 컴파일러에서 "추상 클래스를 예시할 수 없음" 오류를
CAnimal some_animal; // 'CAnimal' 오류 - 컴파일러에서 "추상 클래스를 예시할 수 없음" 오류를

//--- 적절한 사용 예
new CCat;  // 오류 없음 - CCat 클래스가 추상이 아닙니다
CCat cat;  // 오류 없음 - CCat 클래스가 추상이 아닙니다
```

추상 클래스에 대한 제한

© 2000-2025, MetaQuotes Ltd.

추상 클래스 생성자가 (직접 또는 간접적으로) 순수 가상 함수를 호출하는 경우 결과는 정의되지 않습니다.

```
//+------------------------------------------------------------------+
//| 추상 기준 클래스                                                 |
//+------------------------------------------------------------------+
class CAnimal
  {
public:
   //--- 순수 가상 함수
   virtual void      Sound(void)=NULL;
   //--- 함수
   void              CallSound(void) { Sound(); }
   //--- 구성자
   CAnimal()
    {
     //--- 가상 메서드의 명시적 호출
     Sound();
     //--- 암시적 호출(세 번째 함수 사용)
     CallSound();
     //--- 생성자 및/또는 소멸자는 항상 자체 함수를 호출합니다,
     //--- 가상이고 파생 클래스에서 호출된 함수에 의해 재정의된 경우에도
     //--- 호출된 함수가 순수 가상이면
     //--- 이 호출은 심각한 런타임 오류를 발생시킵니다: "순수 가상 함수 호출"
    }
  };
```

그러나 추상 클래스의 생성자와 소멸자는 다른 멤버 함수를 호출할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

네임스페이스

네임스페이스는 변수, 함수, 클래스 등 다양한 ID가 정의되는 특별히 선언된 영역입니다. 이것은 namespace 키워드를 사용함으로써 설정됩니다:

```
namespace name of_space {
  // 함수, 클래스 및 변수 정의 목록
 }
```

'namespace'를 적용하면 전역 네임스페이스를 하위 공간으로 분할할 수 있습니다. 네임스페이스 내의 모든 ID는 규격 없이 서로 사용할 수 있습니다. :: 연산자(컨텍스트 확인 작업)는 외부에서 네임스페이스 멤버에 액세스하는 데 사용됩니다.

```
namespace ProjectData
{
class DataManager
  {
public:
   void              LoadData() {}
  };
void Func(DataManager& manager) {}
 }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ProjectData 네임스페이스로 작업
   ProjectData::DataManager mgr;
   mgr.LoadData();
   ProjectData::Func(mgr);
  }
```

네임스페이스는 논리 그룹 형식으로 코드를 정렬하고 프로그램에서 여러 라이브러리를 사용할 때 발생할 수 있는 이름 충돌을 방지하는 데 사용됩니다. 이러한 경우 각 라이브러리의 네임스페이스에 선언하여 각 라이브러리의 필요한 함수 및 클래스에 명시적으로 액세스할 수 있습니다.

네임스페이스는 하나 이상의 파일에서 여러 블록으로 선언할 수 있습니다. 컴파일러는 사전 처리 중에 모든 부품을 함께 결합하며 결과 네임스페이스에는 모든 부품에 선언된 모든 멤버가 포함됩니다. Sample.mqh에 구현된 A 클래스가 파일을 포함한다고 가정합니다:

```
//+------------------------------------------------------------------+
//|                                                   Sample.mqh |
//+------------------------------------------------------------------+
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
```

© 2000-2025, MetaQuotes Ltd.

우리는 프로젝트에 이 클래스를 사용하고 싶지만, 이미 A 클래스가 있습니다. 클래스를 모두 사용하고 ID 충돌을 방지하려면 네임스페이스에 포함된 파일을 줄 바꿈하면 됩니다:

```
//--- 첫 A 클래스를 선언합니다
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
//--- 충돌을 방지하기 위해 라이브러리 네임스페이스의 Sample.mqh 파일에서 A 클래스를 줄 바꿈
namespace Library
{
#include "Sample.mqh"
 }
//--- 라이브러리 네임스페이스에 다른 클래스 추가
namespace Library
{
class B
  {
public:
                     B() {Print(__FUNCTION__);}
  };
 }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 글로벌 네임스페이스에서 A 클래스 사용
   A a1;
//--- 라이브러리 네임스페이스의 A 및 B 클래스를 사용합니다.
   Library::A a2;
   Library::B b;
  }
//+------------------------------------------------------------------+

/*
결과:
   A::A
   Library::A::A
   Library::B::B
*/
```

네임스페이스는 내포할 수 있습니다. 중첩된 네임스페이스는 상위 공간의 멤버에 대한 무제한 액세스 권한을 가지지만 상위 공간의 멤버에게는 중첩된 네임스페이스에 대한 무제한 액세스 권한이 없습니다.

```
namespace General
{
int Func();
```

© 2000-2025, MetaQuotes Ltd.

```
namespace Details
{
 int Counter;
 int Refresh()  {return Func(); }
 }

int GetBars()   {return(iBars(Symbol(), Period()));};
int Size(int i) {return Details::Counter;}
 }
```

글로벌 네임스페이스

ID가 네임스페이스에 명시적으로 선언되지 않은 경우 글로벌 네임스페이스의 암시적 부분으로 간주됩니다. 글로벌 ID를 명시적으로 설정하려면 이름 없이 범위 지정 연산자를 사용하십시오. 이렇게 하면 이 ID를 다른 네임스페이스에 있는 동일한 이름의 다른 요소와 구별할 수 있습니다. 예를 들어 함수를 가져오는 경우:

```
#import "lib.dll"
int Func();
#import
//+------------------------------------------------------------------+
//|  일부 함수                                                       |
//+------------------------------------------------------------------+
int Func()
  {
   return(0);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//+--- 불러온 함수 호출
   Print(lib::Func());
//+--- 함수를 호출합니다
   Print(::Func());
  }
```

이 경우 DLL 함수에서 가져온 모든 함수가 동일한 이름의 네임스페이스에 포함됩니다. 이를 통해 컴파일러는 호출될 함수를 명확하게 결정할 수 있었습니다.

더 보기

글로벌 변수, 로컬 변수, 가시성 범위 및 변수의 수명, 오브젝트 생성 및 삭제

© 2000-2025, MetaQuotes Ltd.

상수, 열거값 및 구조

프로그램 쓰기를 단순화하고 프로그램 텍스트를 인식에 보다 편리하게 하기 위해 MQL5 언어는 사전 정의된 표준 상수와 열거값을 제공합니다. 그 외에도 서비스 구조를 사용하여 정보를 저장합니다.

표준 상수는 매크로와 유사하며 int 유형입니다.

상수는 다음 용도에 따라 그룹화됩니다:

- 차트 상수는 가격 차트(개장, 탐색, 매개변수 설정)를 다룰 때 사용;
- 개체 상수는 차트에 생성 및 표시할 수 있는 그래픽 개체를 처리하기 위함;
- 지표 상수는 표준 지표 및 사용자 정의 지표 작업에 사용;
- 환경 상태 상수는 MQL5 프로그램의 속성을 설명하고 클라이언트 터미널, 금융 상품 및 현재 계정에 대한 정보를 표시;
- 거래 상수를 통해 트레이딩 과정에서 다양한 정보를 지정가능;
- 명명된 상수는 MQL5 언어의 상수;
- 데이터 구조는 사용된 데이터 스토리지 형식을 설명;
- 오류 및 경고 코드는 거래 요청에 대한 컴파일러 메시지 및 거래 서버 응답을 설명;
- In/out 상수는 파일 기능을 사용하고 MessageBox() 기능에 의해 화면에 메시지를 표시하도록 고안되었습니다.

© 2000-2025, MetaQuotes Ltd.

차트 상수

차트의 다양한 특성을 설명하는 상수는 다음과 같은 그룹으로 나뉩니다:

- 이벤트의 유형 – 차트 작업 시 발생하는 이벤트;
- 차트 시간 프레임 – 표준 내장 기간;
- 차트의 속성 - 차트 함수의 매개변수로 사용되는 식별자;
- 위치지정 상수 - ChartNavigate() 함수의 매개변수 값;
- 차트 표시 – 차트 모양을 설정.

© 2000-2025, MetaQuotes Ltd.

차트 이벤트 유형

사전 정의된 함수 OnChartEvent()를 사용하여 처리할 수 있는 이벤트 유형은 11가지 입니다. 커스텀 이벤트의 경우 65535 식별자가 CHARTEVENT_CUSTOM 에서 CHARTEVENT_CUSTOM_LAST 사이의 범위에서 제공됩니다. 커스텀 이벤트를 생성하려면 EventChartCustom() 함수를 사용해야 합니다.

ENUM_CHART_EVENT

ID	설명
CHARTEVENT_KEYDOWN	키 입력
CHARTEVENT_MOUSE_MOVE	마우스 이동, 마우스 클릭 (CHART_EVENT_MOUSE_MOVE=true 로 차트에 설정된 경우)
CHARTEVENT_MOUSE_WHEEL	마우스 휠을 누르거나 스크롤 (차트에 대해 CHART_EVENT_MOUSE_WHEEL=True 인 경우)
CHARTEVENT_OBJECT_CREATE	그래픽 객체 생성됨 (차트에 대해 CHART_EVENT_OBJECT_CREATE=true 가 설정된 경우)
CHARTEVENT_OBJECT_CHANGE	그래픽 객체 속성이 특성 대화상자를 통해 변경되었습니다
CHARTEVENT_OBJECT_DELETE	그래픽 객체 삭제됨 (CHART_EVENT_OBJECT_DELETE=true 가 차트에 대해 설정된 경우)
CHARTEVENT_CLICK	차트에 클릭
CHARTEVENT_OBJECT_CLICK	다음에 대해 클릭: 그래픽 객체
CHARTEVENT_OBJECT_DRAG	드래그 및 드롭: 그래픽 객체
CHARTEVENT_OBJECT_ENDEDIT	그래픽 객체 편집의 텍스트 편집 끝
CHARTEVENT_CHART_CHANGE	특성 대화상자를 통한 차트 크기 변경 또는 차트 속성 수정
CHARTEVENT_CUSTOM	커스텀 이벤트 범위의 초기 이벤트 수
CHARTEVENT_CUSTOM_LAST	커스텀 이벤트 범위에서 이벤트의 최종 번호

각 이벤트 유형에 대해 OnChartEvent() 함수의 입력 매개 변수에는 이 이벤트 처리에 필요한 한정된 값이 있습니다. 이 매개 변수를 통과하는 이벤트 및 값은 아래 표에 나열되어 있습니다.

Event	id 매개 변수 값	lparam 매개변수 값	dparam 매개변수 값	sparam 매개변수 값
키 입력 이벤트	CHARTEVENT_KEYDOWN	누른 키의 코드	반복 횟수(사용자가 키를 누른 결과로 키 입력이 반복되는 횟수)	키보드 버튼의 상태를 설명하는 비트 마스크의 문자열 값

© 2000-2025, MetaQuotes Ltd.

Event	id 매개 변수 값	lparam 매개변수 값	dparam 매개변수 값	sparam 매개변수 값
마우스 이벤트 (차트에 대해 CHART_EVENT_MOUSE_MOVE=true 가 설정된 경우)	CHARTEVENT_MOUSE_MOVE	X 좌표	Y 좌표	마우스 버튼의 상태를 설명하는 비트 마스크의 문자열 값
마우스 휠 이벤트 (차트에 대해 CHART_EVENT_MOUSE_WHEEL=true 인 경우)	CHARTEVENT_MOUSE_WHEEL	마우스 포인터의 X 및 Y 좌표와 키 및 마우스 버튼의 상태 플래그. 다음을 참조하십시오: 아래 예제	마우스 휠 스크롤의 델타 값	–
그래픽 객체 생성 이벤트 (차트에 대해 CHART_EVENT_OBJECT_CREATE=true 로 설정된 경우)	CHARTEVENT_OBJECT_CREATE	–	–	생성된 그래픽 개체의 이름
속성 대화상자를 통한 객체 속성 변경 이벤트	CHARTEVENT_OBJECT_CHANGE	–	–	수정된 그래픽 개체의 이름
그래픽 객체 삭제 이벤트 (CHART_EVENT_OBJECT_DELETE =true 가 차트에 대해 설정된 경우)	CHARTEVENT_OBJECT_DELETE	–	–	삭제된 그래픽 개체의 이름
차트를 마우스로 클릭하는 이벤트	CHARTEVENT_CLICK	X 좌표	Y 좌표	–
차트에 속한 그래픽 개체를 마우스로 클릭하는 이벤트	CHARTEVENT_OBJECT_CLICK	X 좌표	Y 좌표	이벤트가 발생한 그래픽 개체의 이름
마우스를 사용하여 그래픽 개체를 드래그하는 이벤트	CHARTEVENT_OBJECT_DRAG	–	–	이동된 그래픽 개체의 이름
레이블 편집 그래픽 객체의 입력 상자에서 완료된	CHARTEVENT_OBJECT_ENDEDIT	–	–	다음의 이름: 텍스트 편집이 완료된 LabelEdit 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Event	id 매개 변수 값	lparam 매개변수 값	dparam 매개변수 값	sparam 매개변수 값
텍스트 편집 이벤트				
변경 이벤트 – 특성 대화상자를 통한 차트 크기 또는 차트 속성 수정에 대한	CHARTEVENT_CHART_CHANGE	–	–	–
N 넘버 이하의 유저 이벤트 ID	CHARTEVENT_CUSTOM+N	EventChartCustom() 함수에 의해 설정된 값	EventChartCustom() 함수에 의해 설정된 값	EventChartCustom() 함수에 의해 설정된 값

예제:

```
#define KEY_NUMPAD_5        12
#define KEY_LEFT            37
#define KEY_UP              38
#define KEY_RIGHT           39
#define KEY_DOWN            40
#define KEY_NUMLOCK_DOWN    98
#define KEY_NUMLOCK_LEFT   100
#define KEY_NUMLOCK_5      101
#define KEY_NUMLOCK_RIGHT  102
#define KEY_NUMLOCK_UP     104
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("이름이 ",MQL5InfoString(MQL5_PROGRAM_NAME)," 인 expert가 실행 중입니다");
//--- 객체 생성 이벤트 활성화
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- 객체 삭제 이벤트 활성화
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- 차트 속성의 강제 업데이트로 이벤트 처리 준비 상태 보장
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent 함수                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 식별자
                  const long& lparam,   // long 형식의 이벤트 매개 변수
                  const double& dparam, // double 형식의 이벤트 매개 변수
```

© 2000-2025, MetaQuotes Ltd.

```
                  const string& sparam  // string 형식의 이벤트 매개 변수
                  )
  {
//--- 차트에서 마우스 왼쪽 버튼을 눌렀습니다
   if(id==CHARTEVENT_CLICK)
     {
      Print("차트에서 마우스 클릭의 좌표는 다음과 같습니다: x = ",lparam,"  y = ",dparam);
     }
//--- 그래픽 객체를 마우스로 클릭했습니다
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("다음 이름의 객체를 마우스로 클릭했습니다 '"+sparam+"'");
     }
//--- 키 눌림
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("KEY_NUMLOCK_LEFT 이(가) 눌렸습니다");   break;
         case KEY_LEFT:          Print("KEY_LEFT 이(가) 눌렸습니다");           break;
         case KEY_NUMLOCK_UP:    Print("KEY_NUMLOCK_UP 이(가) 눌렸습니다");     break;
         case KEY_UP:            Print("KEY_UP 이(가) 눌렸습니다");             break;
         case KEY_NUMLOCK_RIGHT: Print("KEY_NUMLOCK_RIGHT 이(가) 눌렸습니다");  break;
         case KEY_RIGHT:         Print("KEY_RIGHT 이(가) 눌렸습니다");          break;
         case KEY_NUMLOCK_DOWN:  Print("KEY_NUMLOCK_DOWN 이(가) 눌렸습니다");   break;
         case KEY_DOWN:          Print("KEY_DOWN 이(가) 눌렸습니다");           break;
         case KEY_NUMPAD_5:      Print("KEY_NUMPAD_5 이(가) 눌렸습니다");       break;
         case KEY_NUMLOCK_5:     Print("KEY_NUMLOCK_5 이(가) 눌렸습니다");      break;
         default:                Print("목록에 없는 일부 키가 눌렸습니다");
        }
      ChartRedraw();
     }
//--- 개체가 삭제되었습니다
   if(id==CHARTEVENT_OBJECT_DELETE)
     {
      Print(" ",sparam," 의 이름의 객체가 삭제되었습니다");
     }
//--- 객체가 생성되었습니다
   if(id==CHARTEVENT_OBJECT_CREATE)
     {
      Print(" ",sparam," 의 이름의 객체가 생성되었습니다");
     }
//--- 객체가 이동되었거나 고정점 좌표가 변경되었습니다
   if(id==CHARTEVENT_OBJECT_DRAG)
     {
      Print("이름이 ",sparam," 인 객체의 고정점 좌표가 변경되었습니다");
     }
//--- 개체 편집의 텍스트가 변경되었습니다
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      Print("이름이 ",sparam," 인 객체의 편집 필드에 있는 텍스트가 변경되었습니다");
     }
  }
```

CHARTEVENT_MOUSE_MOVE 이벤트의 경우 **sparam** 문자열 매개변수는 키보드 및 마우스 버튼의 상태에 대한 정보를 포함합니다:

Bit	설명
1	마우스 좌측 버튼 상태
2	마우스 우측 버튼 상태
3	SHIFT 버튼 상태
4	CTRL 버튼 상태
5	마우스 중간 버튼 상태
6	마우스 첫 여분 버튼 상태
7	마우스 둘째 여분 버튼 상태

예제:

```
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- CHART_EVENT_MOUSE_MOVE 메시지 활성화
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- 차트의 상황에 맞는 메뉴 사용 안 함(오른쪽)
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
//--- 십자선 비활성화(가운데 버튼 사용)
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
//--- 차트 속성의 강제 업데이트로 이벤트 처리 준비 상태 보장
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse left
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse right
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse middle
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse first X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse second X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
```

© 2000-2025, MetaQuotes Ltd.

```
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent 함수                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

CHARTEVENT_MOUSE_WHEEL 이벤트의 경우 파라미터 **lparam** 및 **dparam** 에는 마우스 버튼, 커서 좌표 및 마우스 휠 스크롤 값의 **Ctrl** 및 **Shift** 키 상태에 대한 정보가 포함됩니다.더 알아보고 싶으시다면 차트에서 이 Expert Advisor를 실행하고 마우스 휠을 스크롤한 상태에서 다른 버튼을 누르고 코드에 설명된 키를 누르십시오.

예제 - CHARTEVENT_MOUSE_WHEEL 이벤트 처리:

```
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
init OnInit()
  {
//--- 마우스 휠 스크롤 메시지 사용
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- 차트 속성의 강제 업데이트를 통해 이벤트 처리 준비 상태 보장
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent 함수                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- 이 이벤트에 대해 마우스 버튼 및 휠의 상태를 고려하십시오
      int flg_keys = (int)(lparam>>32);          // Ctrl 및 Shift 키의 상태 플래그 및 마
      int x_cursor = (int)(short)lparam;         // 마우스 휠 이벤트가 발생한 X 좌표
      int y_cursor = (int)(short)(lparam>>16);   // 마우스 휠 이벤트가 발생한 Y 좌표
      int delta    = (int)dparam;                // 마우스 스크롤의 총 값, +120 또는 -120
      //--- flag 처리
      string str_keys="";
      if((flg_keys&0x0001)!=0) str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0) str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0) str_keys+="SHIFT ";
```

© 2000-2025, MetaQuotes Ltd.

```
      if((flg_keys&0x0008)!=0) str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0) str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0) str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0) str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1) + "'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
    }
  }
//+------------------------------------------------------------------+ /*
   결과 예제
   CHARTEVENT_MOUSE_WHEEL: Ctrl pressed: X=193, Y=445, delta=-120
   CHARTEVENT_MOUSE_WHEEL: Shift pressed: X=186, Y=446, delta=120
   CHARTEVENT_MOUSE_WHEEL:  X=178, Y=447, delta=-120
   CHARTEVENT_MOUSE_WHEEL:  X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: MiddleButton pressed: X=231, Y=449, delta=120
   CHARTEVENT_MOUSE_WHEEL: LeftButton pressed: X=279, Y=320, delta=-120
   CHARTEVENT_MOUSE_WHEEL: RightButton pressed: X=253, Y=330, delta=120  */
```

더 보기

이벤트 핸들링 함수, 이벤트 작업

© 2000-2025, MetaQuotes Ltd.

차트 타임프레임

차트의 미리 정의된 모든 시간 범위에는 고유 식별자가 있습니다. PERIOD_CURRENT 식별자는 mql5-프로그램이 실행 중인 차트의 현재 주기를 의미합니다.

ENUM_TIMEFRAMES

ID	설명
PERIOD_CURRENT	현재 타임프레임
PERIOD_M1	1분
PERIOD_M2	2분
PERIOD_M3	3분
PERIOD_M4	4분
PERIOD_M5	5분
PERIOD_M6	6분
PERIOD_M10	10분
PERIOD_M12	12분
PERIOD_M15	15분
PERIOD_M20	20분
PERIOD_M30	30분
PERIOD_H1	1시간
PERIOD_H2	2시간
PERIOD_H3	3시간
PERIOD_H4	4시간
PERIOD_H6	6시간
PERIOD_H8	8시간
PERIOD_H12	12시간
PERIOD_D1	1일
PERIOD_W1	1주
PERIOD_MN1	1달

예제:

```
   string chart_name="test_Object_Chart";
   Print("이름을 사용하여 차트 객체를 생성해 보십시오 ",chart_name);
//--- 해당 객체가 없는 경우 - 생성하십시오
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 심볼 정의
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- 고정점의 X 좌표 설정
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- 고정점의 Y 좌표 설정
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- 차트 너비 설정
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- 차트 높이 설정
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- 타임프레임 설정
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- 스케일 설정(0에서 5까지)
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- 마우스로 선택 해제
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

시계열 식별자

시계열 식별자는 iHighest() 및 iLowest() 함수에 사용됩니다. 열거값과 같을 수 있습니다

ENUM_SERIESMODE

식별자	설명
MODE_OPEN	시가
MODE_LOW	저가
MODE_HIGH	고가
MODE_CLOSE	종가
MODE_VOLUME	틱 볼륨
MODE_REAL_VOLUME	리얼 볼륨
MODE_SPREAD	스프레드

더 보기

PeriodSeconds, 기간, 날짜 및 시간, 객체 가시성

© 2000-2025, MetaQuotes Ltd.

차트 속성

ENUM_CHART_PROPERTY 열거형 식별자는 차트로 작업하기 위한 함수의 매개 변수로 사용됩니다. "속성 유형" 칼럼의 r/o 약어는 이 속성이 읽기 전용이며 변경할 수 없음을 의미합니다./ "속성 유형" 칼럼의 w/o 약어는 이 속성이 쓰기 전용이며 수신할 수 없음을 의미합니다. 특정 속성에 액세스할 때는 차트 하위 창의 수를 나타내는 추가 매개 변수 수정자(한정자)를 지정해야 합니다. 0은 메인 창을 의미합니다

차트 속성을 정의하는 함수는 실제로 차트에 변경 명령을 보내는 데 사용됩니다. 이러한 함수가 성공적으로 실행되면 명령이 차트 이벤트의 공통 대기열에 포함됩니다. 차트 이벤트의 대기열을 처리할 때 변경 사항이 차트에 구현됩니다.

따라서 이러한 기능을 호출한 후 차트의 즉각적인 시각적 업데이트를 기대하지 마십시오. 일반적으로 차트는 새 견적 도착, 차트 창 크기 조정 등 변경 이벤트 이후에 터미널에 의해 자동으로 업데이트됩니다. ChartRedraw() 함수를 사용하여 차트를 강제로 업데이트합니다.

다음 함수의 경우: ChartSetInteger() 및 ChartGetInteger()

ENUM_CHART_PROPERTY_INTEGER

ID	설명	속성 유형
CHART_SHOW	가격 차트 그리기. 거짓이면 가격 차트 속성을 그릴 수 없고 시간 및 가격 척도, 빠른 탐색 막대, 달력 이벤트 레이블, 지표 및 막대 도구 설명, 지표 하위 창, 볼륨 히스토그램 등 모든 차트 테두리 렌더가 제거됩니다. 도면을 비활성화하면 그래픽 리소스를 사용하여 사용자 정의 프로그램 인터페이스를 만들 수 있는 완벽한 솔루션입니다. 그래픽 객체 는 CHART_SHOW 특성 값에 관계없이 항상 그려집니다.	bool
CHART_IS_OBJECT	"차트"(OBJ_CHART) 개체 식별 – 그래픽 개체에 대해 true를 반환합니다. 실제 차트에 대해 false를 반환합니다	bool 읽기전용
CHART_BRING_TO_TOP	다른 차트 맨 위에 차트 표시	bool
CHART_CONTEXT_MENU	마우스 오른쪽 버튼을 사용하여 상황에 맞는 메뉴에 대한 액세스 사용/사용 안 함. CHART_CONTEXT_MENU=false 면, 차트의 상황에 맞는 메뉴만 비활성화됩니다. 차트에 있는 개체의 상황에 맞는 메뉴를 사용할 수 있습니다.	bool (디폴트는 true)
CHART_CROSSHAIR_TOOL	가운데 클릭을 사용하여 십자형 도구에 대한 액세스를 활성화/비활성화합니다.	bool (디폴트는 true)
CHART_MOUSE_SCROLL	마우스 왼쪽 버튼을 사용하여 차트를 수평으로 스크롤합니다. 또한 수직 스크롤은 다음 속성 값이 참으로 설정된 경우에도 사용할 수 있습니다: CHART_SCALEFIX,	bool

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
	CHART_SCALEFIX_11 또는 CHART_SCALE_PT_PER_BAR CHART_MOUSE_SCROLL=false 면, 마우스 휠을 사용한 차트 스크롤을 사용할 수 없습니다	
CHART_EVENT_MOUSE_WHEEL	차트의 모든 mql5 프로그램에 마우스 휠 이벤트(CHARTEVENT_MOUSE_WHEEL)에 대한 메시지 보내기	bool (디폴트는 true)
CHART_EVENT_MOUSE_MOVE	차트의 모든 mql5 프로그램에 마우스 이동 및 마우스 클릭 이벤트(CHARTEVENT_MOUSE_MOVE)에 대한 알림 보내기	bool
CHART_EVENT_OBJECT_CREATE	차트의 모든 mql5-프로그램에 새 개체 생성(CHARTEVENT_OBJECT_CREATE) 이벤트 알림 보내기	bool
CHART_EVENT_OBJECT_DELETE	차트의 모든 mql5-프로그램에 개체 삭제 이벤트(CHARTEVENT_OBJECT_DELETE)에 대한 알림	bool
CHART_MODE	차트 유형(캔들스틱, 막대 또는 선)	enum ENUM_CHART_MODE
CHART_FOREGROUND	전경의 가격 차트	bool
CHART_SHIFT	오른쪽 테두리에서 들여쓰는 가격 차트 모드	bool
CHART_AUTOSCROLL	차트의 오른쪽 테두리로 자동 이동 모드	bool
CHART_KEYBOARD_CONTROL	키보드("홈", "끝", "페이지업", "+", "-", "위쪽 화살표" 등)를 사용하여 차트를 관리할 수 있습니다. CHART_KEYBOARD_CONTROL를 false로 설정하면 차트 스크롤 및 스케일링이 비활성화되며 OnChartEvent()에서 이벤트를 누르는 키를 수신할 수 있습니다.	bool
CHART_QUICK_NAVIGATION	차트가 Space를 가로채고 Enter 키를 눌러 빠른 탐색 막대를 활성화합니다. 마우스를 두 번 클릭하거나 Space/Enter를 누르면 차트 하단에 빠른 탐색 막대가 자동으로 나타납니다. 이를 통해 기호, 시간 범위 및 처음 표시되는 막대 날짜를 빠르게 변경할 수 있습니다.	bool
CHART_SCALE	스케일	int 0에서부터 5까지
CHART_SCALEFIX	고정 스케일 모드	bool
CHART_SCALEFIX_11	스케일 1:1 모드	bool

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
CHART_SCALE_PT_PER_BAR	막대당 포인트로 지정할 축척도	bool
CHART_SHOW_TICKER	왼쪽 상단 모서리에 기호 스티커를 표시합니다. CHART_SHOW_TICKER 를 'false'로 설정하면 CHART_SHOW_OHLC도 'false'로 설정되고 OHLC가 비활성화됩니다	bool
CHART_SHOW_OHLC	왼쪽 상단에 OHLC 값을 표시합니다. CHART_SHOW_OHLC 를 'true'로 설정하면 CHART_SHOW_TICKER도 'true'로 설정되고 티커가 활성화됩니다	bool
CHART_SHOW_BID_LINE	차트에서 입찰 값을 수평선으로 표시	bool
CHART_SHOW_ASK_LINE	차트에서 묻기 값을 수평선으로 표시	bool
CHART_SHOW_LAST_LINE	차트에서 마지막 값을 수평선으로 표시	bool
CHART_SHOW_PERIOD_SEP	인접 기간 사이의 수직 구분 기호 표시	bool
CHART_SHOW_GRID	차트에 그리드 표시	bool
CHART_SHOW_VOLUMES	차트에 볼륨 표시	enum ENUM_CHART_VOLUME_MODE
CHART_SHOW_OBJECT_DESCR	객체에 대한 텍스트 설명 표시(일부 객체에 사용할 수 없음)	bool
CHART_SHOW_TRADE_HISTORY	거래 내역의 거래를 진입/청산의 화살표로 차트에 표시합니다. 플랫폼 설정의 옵션 설명에서 거래 내역 보기""를 참조하십시오.	bool
CHART_VISIBLE_BARS	차트에서 표시할 수 있는 막대 수	int 읽기전용
CHART_WINDOWS_TOTAL	지표 하위 창을 포함한 총 차트 창 수	int 읽기전용
CHART_WINDOW_IS_VISIBLE	하위 창의 가시성	bool 읽기전용 한정자 – 하위 창 번호
CHART_WINDOW_HANDLE	차트 창 핸들 (HWND)	int 읽기전용
CHART_WINDOW_YDISTANCE	세로 Y축을 따라 표시기 하위 창의 위쪽 프레임과 기본 차트 창의 위쪽 프레임 사이의 거리(픽셀). 마우스 이벤트의 경우, 커서 좌표는 기본 차트 창의 좌표로 전달되며, 지표	int 읽기전용 한정자 – 하위 창 번호

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
	하위 창의 그래픽 객체의 좌표는 하위 창의 왼쪽 상단 모서리를 기준으로 설정됩니다. 이 값은 그래픽 객체에 대한 올바른 작업을 위해 기본 차트의 절대 좌표를 하위 창의 로컬 좌표로 변환하는 데 필요합니다. 그래픽 객체의 좌표는 하위 창 프레임의 왼쪽 상단 모서리에 상대적으로 설정됩니다.	
CHART_FIRST_VISIBLE_BAR	차트에서 첫 번째로 표시되는 막대의 수. 막대의 인덱싱이 시계열의 인덱싱과 동일합니다.	int 읽기전용
CHART_WIDTH_IN_BARS	막대의 차트 너비	int 읽기전용
CHART_WIDTH_IN_PIXELS	차트 너비(픽셀)	int 읽기전용
CHART_HEIGHT_IN_PIXELS	차트 높이(픽셀)	int 한정자 –하위 창 번호
CHART_COLOR_BACKGROUND	차트 배경색	색상
CHART_COLOR_FOREGROUND	축, 스케일, OHLC 선 색상	색상
CHART_COLOR_GRID	그리드 색상	색상
CHART_COLOR_VOLUME	볼륨 및 포지션 레벨의 색상	색상
CHART_COLOR_CHART_UP	강세 촛대의 위쪽 막대, 그림자 및 차체 테두리에 대한 색상	색상
CHART_COLOR_CHART_DOWN	강세 촛대의 아래쪽 막대, 그림자 및 차체 테두리에 대한 색상	색상
CHART_COLOR_CHART_LINE	도지 일본 캔들스틱의 선 차트 색상 및 색상	색상
CHART_COLOR_CANDLE_BULL	강세 캔들스틱의 본체 색상	색상
CHART_COLOR_CANDLE_BEAR	약세 캔들스틱의 본체 색상	색상
CHART_COLOR_BID	입찰가 수준 색상	색상
CHART_COLOR_ASK	요청가 수준 색상	색상
CHART_COLOR_LAST	마지막으로 실행된 거래 가격(종가)의 라인 색상	색상

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
CHART_COLOR_STOP_LEVEL	스탑 오더 레벨 색상(스탑 로스 및 이익 실현)	색상
CHART_SHOW_TRADE_LEVELS	차트에 거래 수준 표시(공개 포지션 수준, 스탑 로스, 이익 실현 및 보류 중인 주문)	bool
CHART_DRAG_TRADE_LEVELS	마우스로 차트의 거래 수준을 끌 수 있는 권한입니다. 드래그 모드는 기본적으로 활성화되어 있습니다(참값).	bool
CHART_SHOW_DATE_SCALE	차트에 시간 척도 표시	bool
CHART_SHOW_PRICE_SCALE	차트에 가격 척도 표시	bool
CHART_SHOW_ONE_CLICK	표시: "원클릭 트레이딩"차트의 패널	bool
CHART_IS_MAXIMIZED	차트 창이 최대화됩니다	bool r/o
CHART_IS_MINIMIZED	차트 창이 최소화됩니다	bool r/o
CHART_IS_DOCKED	차트 창이 도킹되었습니다. false로 설정하면 차트를 터미널 영역 외부로 끌 수 있습니다	bool
CHART_FLOAT_LEFT	가상 화면을 기준으로 도킹 해제된 차트 창의 왼쪽 좌표	int
CHART_FLOAT_TOP	가상 화면에 상대적인 도킹 해제된 차트 창의 위쪽 좌표	int
CHART_FLOAT_RIGHT	가상 화면에 상대적인 도킹 해제된 차트 창의 오른쪽 좌표	int
CHART_FLOAT_BOTTOM	가상 화면에 상대적인 도킹 해제된 차트 창의 아래쪽 좌표	int

다음 함수의 경우: ChartSetDouble() 및 ChartGetDouble()

ENUM_CHART_PROPERTY_DOUBLE

ID	설명	속성 유형
CHART_SHIFT_SIZE	오른쪽 테두리에서 들여쓰는 0 막대 크기(%)	2배(10 ~50%)
CHART_FIXED_POSITION	왼쪽 테두리부터 고정된 위치를 백분율 값으로 차트 화합니다. 차트 고정 위치는 수평 시	double

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
	간 축에 작은 회색 삼각형으로 표시됩니다. 들어오는 틱에서 오른쪽으로 스크롤하는 자동 차트가 비활성화된 경우에만 표시됩니다 (CHART_AUTOSCROLL 속성 참조). 확대 및 축소할 때 고정 위치의 막대가 동일한 위치에 유지됩니다.	
CHART_FIXED_MAX	차트 최대화 고정	double
CHART_FIXED_MIN	차트 최소화 고정	double
CHART_POINTS_PER_BAR	막대 당 포인트 척도	double
CHART_PRICE_MIN	차트 최소화	double 읽기전용 한정자 – 하위 창 번호
CHART_PRICE_MAX	차트 최대화	double 읽기전용 한정자 – 하위 창 번호

다음 함수용: ChartSetString() 및 ChartGetString()

ENUM_CHART_PROPERTY_STRING

ID	설명	속성 유형
CHART_COMMENT	차트의 주석 텍스트	string
CHART_EXPERT_NAME	지정된 chart_id로 차트에서 실행 중인 Expert Advisor의 이름	string 읽기전용
CHART_SCRIPT_NAME	차트에서 지정된 chart_id로 실행 중인 스크립트의 이름	string 읽기전용

예제:

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
  {
   case(CHART_BARS):    Print("CHART_BARS");   break;
   case(CHART_CANDLES): Print("CHART_CANDLES");break;
   default:Print("CHART_LINE");
```

© 2000-2025, MetaQuotes Ltd.

```
  }
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT = true");
else Print("CHART_SHIFT = false");
bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
if(autoscroll) Print("CHART_AUTOSCROLL = true");
else Print("CHART_AUTOSCROLL = false");
int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
Print("CHART_WINDOW_HANDLE = ",chartHandle);
int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("CHART_WINDOWS_TOTAL = ",windows);
if(windows>1)
  {
   for(int i=0;i<windows;i++)
     {
      int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
      double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
      double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
      Print(i+": CHART_HEIGHT_IN_PIXELS = ",height," pixels");
      Print(i+": CHART_PRICE_MIN = ",priceMin);
      Print(i+": CHART_PRICE_MAX = ",priceMax);
     }
  }
```

더 보기

차트 작업 예제

© 2000-2025, MetaQuotes Ltd.

포지셔닝 상수

ENUM_CHART_POSITION 목록의 세 가지 식별자는 *포지션* 파라미터 값으로, ChartNavigate() 함수에 사용할 수 있습니다.

ENUM_CHART_POSITION

ID	설명
CHART_BEGIN	차트 시작 (가장 오래된 가격)
CHART_CURRENT_POS	현재 포지션
CHART_END	Chart end (the latest prices)

예제:

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
  {
   ChartSetInteger(handle,CHART_AUTOSCROLL,false);
   ChartSetInteger(handle,CHART_SHIFT,true);
   ChartSetInteger(handle,CHART_MODE,CHART_LINE);
   ResetLastError();
   bool res=ChartNavigate(handle,CHART_END,150);
   if(!res) Print("Navigate failed. Error = ",GetLastError());
   ChartRedraw();
  }
```

© 2000-2025, MetaQuotes Ltd.

차트 표현

가격 차트는 다음 세 가지 방법으로 표시할 수 있습니다.

- 막대기로;
- 캔들스틱으로;
- 라인으로.

가격 차트를 표시하는 특정 방법은 함수 ChartSetInteger(chart_handle,CHART_MODE, chart_mode)로 설정됩니다. 여기서 chart_mode 는 ENUM_CHART_MODE 열거값 중 하나입니다.

ENUM_CHART_MODE

ID	설명
CHART_BARS	일련의 막대로 표시
CHART_CANDLES	일본 캔들스틱으로 표시
CHART_LINE	마감 가격으로 그려진 선으로 표시

가격 차트에서 볼륨을 표시하는 모드를 지정하려면 함수 ChartSetInteger(chart_handle, CHART_SHOW_VOLUMES, volume_mode)를 사용합니다. 여기서 volume_mode 는 ENUM_CHART_VOLUME_MODE 열거값 중 하나입니다.

ENUM_CHART_VOLUME_MODE

ID	설명
CHART_VOLUME_HIDE	볼륨이 표시되지 않음
CHART_VOLUME_TICK	틱 볼륨
CHART_VOLUME_REAL	거래량

예제:

```
//--- 현재 차트의 핸들을 가져옵니다
   long handle=ChartID();
   if(handle>0) // 성공한 경우 추가 커스텀
     {
      //--- 자동스크롤 비활성화
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- 차트의 오른쪽 테두리 들여쓰기 설정
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- 캔들스틱으로 표시
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- 히스토리가 시작될 때부터 100bar 씩 스크롤합니다
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- 틱 볼륨 표시 모드 설정
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
     }
```

© 2000-2025, MetaQuotes Ltd.

더 보기

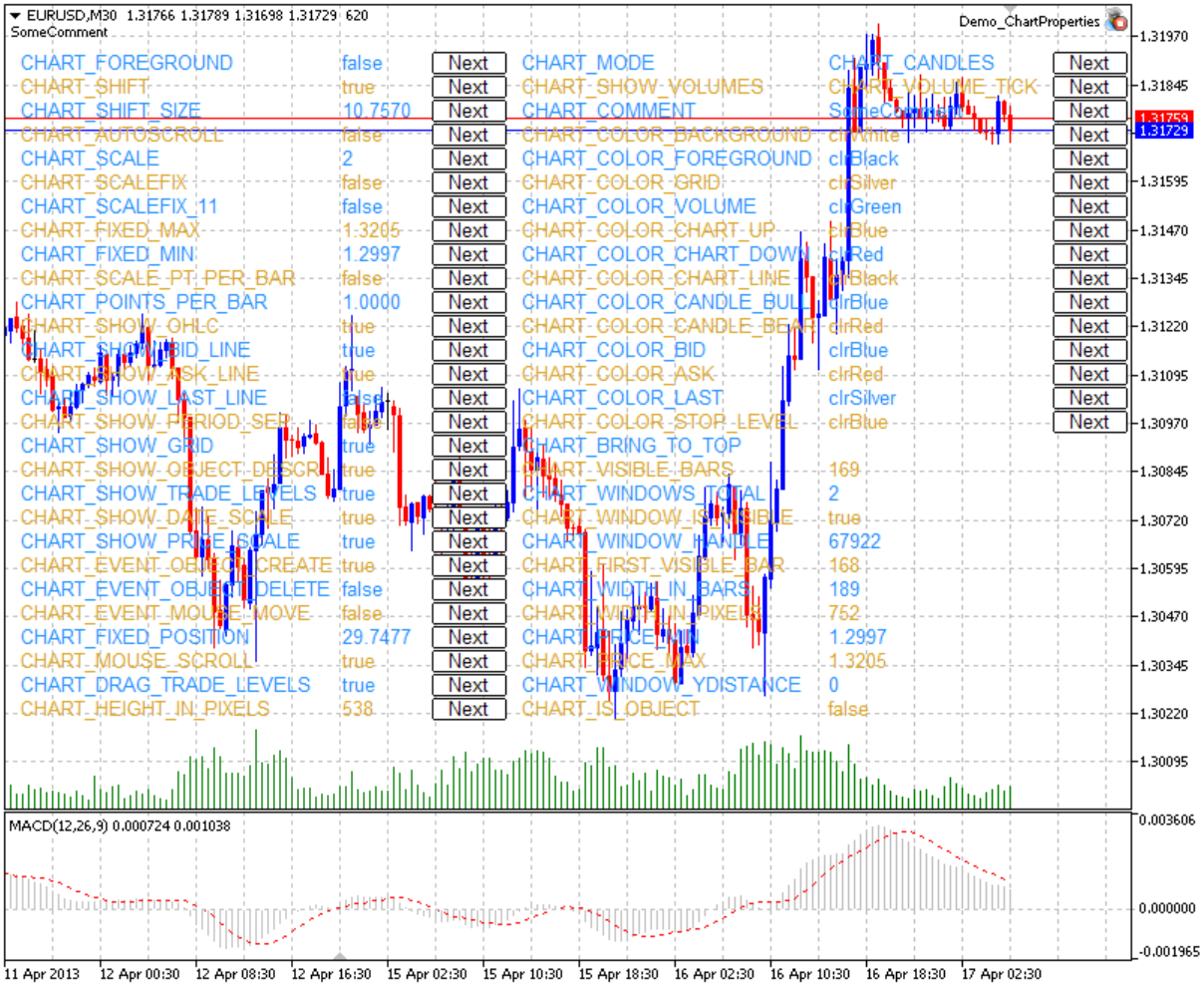
ChartOpen, ChartID

© 2000–2025, MetaQuotes Ltd.

차트 작업 예제

이 절에는 차트 속성으로 작업하는 예가 나와 있습니다. 각 속성에 대해 하나 또는 두 개의 완전한 함수가 표시됩니다. 이러한 기능을 통해 속성 값을 설정/수신할 수 있습니다. 이러한 함수는 커스텀 mql5 애플리케이션에서 "있는 그대로" 사용할 수 있습니다.

아래 스크린샷은 차트 속성의 변경사항이 어떻게 모양을 변화시키는지를 보여주는 그래픽 패널을 보여줍니다. Next 버튼을 클릭하면 적절한 속성의 새 값을 설정하고 차트 창에서 변경 사항을 볼 수 있습니다.



패널의 소스 코드는 아래에 있습니다.

차트 특성 및 샘플 함수 관련 작업

- **CHART_IS_OBJECT** 는 객체가 실제 차트인지 그래픽 객체인지 정의합니다.

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 객체가 차트인지 확인합니다. 만약 그것이 그래픽 객체라면,        |
//| 결과는 참입니다. 그것이 실제 차트라면, 결과 변수는    |
//| 거짓의 값을 가집니다.                                            |
//+------------------------------------------------------------------+
bool ChartIsObject(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 차트 속성 가져오기
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      //--- false 반환
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
```

- **CHART_BRING_TO_TOP**는 차트를 다른 차트 위에 표시합니다.

```
//+------------------------------------------------------------------+
//| 터미널에 명령을 보내 다른 모든 차트보다 차트를 표시합니다  |
//+------------------------------------------------------------------+
bool ChartBringToTop(const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 차트를 다른 차트 위에 표시하다
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_MOUSE_SCROLL**는 마우스 왼쪽 버튼을 사용하여 차트를 스크롤하는 속성입니다.

```
//+------------------------------------------------------------------+
//| 마우스 왼쪽 버튼을 사용하여 차트 스크롤이 활성화되어 있는지 확인합니다          |
//+------------------------------------------------------------------+
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 마우스 왼쪽 버튼을 사용하여 차트 스크롤을 활성화/비활성화합니다        |
//+------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_EVENT_MOUSE_MOVE** 는 mql5 어플리케이션(CHARTEVENT_MOUSE_MOVE)으로 이동 이벤트 및 마우스 클릭과 관련된 메시지를 보내는 속성입니다.

```
//+------------------------------------------------------------------+
//| 이동 이벤트 및 마우스 클릭과 관련된 메시지 확인                    |
//| 차트의 모든 MQL5 어플리케이션에 전송됩니다                         |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------------------+
//| 이동 이벤트와 관련된 메시지 전송 모드를 활성화/비활성화합니다      |
//| 차트에서 MQL5 애플리케이션을 마우스 클릭                                    |
//+------------------------------------------------------------------------------+
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE**는 그래픽 객체 생성 이벤트와 관련된 메시지를 mql5 응용 프로그램 (CHARTEVENT_OBJECT_CREATE)에 보내는 속성입니다.

```
//+------------------------------------------------------------------+
//| 그래픽 이벤트와 관련된 메시지 확인                  |
//| 개체 생성은 차트의 모든 MQL5 응용 프로그램에 전송됩니다      |
//+------------------------------------------------------------------+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------------+
//| 차트에 있는 모든 mql5 응용 프로그램에 그래픽 개체 생성 이벤트와  |
//| 관련된 메시지를 보내는 모드를 활성화/비활성화합니다             |
//+------------------------------------------------------------------------+
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE**는 그래픽 객체 삭제 이벤트와 관련된 메시지를 mql5 응용 프로그램 (CHARTEVENT_OBJECT_DELETE)에 보내는 속성입니다.

```
//+------------------------------------------------------------------+
//| 그래픽 객체 삭제 이벤트와 관련된 메시지가 차트의          |
//| 모든 mql5 응용 프로그램에 전송되는지 확인합니다             |
//+------------------------------------------------------------------+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 전문가 저널에 오류 메시지를 표시합니다
       Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
       return(false);
      }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------------+
//| 차트에 있는 모든 mql5 응용 프로그램에 그래픽 개체 생성 이벤트와  |
//| 차트에 있는 모든 mql5 응용 프로그램에 대한 그래픽 개체 삭제            |
//+------------------------------------------------------------------------+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_MODE** - 차트 유형 (캔들스틱, 막대 또는 선).

```
//+------------------------------------------------------------------+
//| 차트 표시 유형(촛대, 막대 또는 선)을 가져옵니다             |
//+------------------------------------------------------------------+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=WRONG_VALUE;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((ENUM_CHART_MODE)result);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 차트 표시 유형 설정(캔들스틱, 막대 또는 선)                      |
//+------------------------------------------------------------------+
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_FOREGROUND**는 가격 차트를 맨 앞에 표시하는 속성입니다.

```
//+------------------------------------------------------------------+
//| 가격 차트가 전면에 표시되는지 확인합니다                         |
//+------------------------------------------------------------------+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 전면에 가격 차트를 표시하거나 표시하지 않습니다                  |
//+------------------------------------------------------------------+
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHIFT** - 오른쪽 경계에서 가격 차트의 이동 모드.

```
//+------------------------------------------------------------------+
//| 오른쪽 테두리에서 가격 차트를 이동할 수 있는지 확인합니다 |
//+------------------------------------------------------------------+
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------------------+
//| 오른쪽 테두리에서 전환하여 가격 차트를 표시하거나 표시하지 않습니다 |
//+------------------------------------------------------------------------------+
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_AUTOSCROLL** - 차트의 오른쪽 테두리로 자동 이동 모드.

```
//+------------------------------------------------------------------+
//| 새 틱 도착 시 차트의 오른쪽 자동 스크롤이            |
//| 활성화되어 있는지 확인합니다                                     |
//+------------------------------------------------------------------+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 새 틱 도착 시 차트의 오른쪽 자동 스크롤을     |
//| 활설화/비활성화합니다                                            |
//+------------------------------------------------------------------+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
```

- **CHART_SCALE** - 차트 스케일 속성.

```
//+------------------------------------------------------------------+
//| 차트 스케일 가져오기 (0에서 5까지)                                |
//+------------------------------------------------------------------+
int ChartScaleGet(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
//+------------------------------------------------------------------+
//| 차트 스케일 설정 (0에서 5까지)                                    |
//+------------------------------------------------------------------+
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SCALEFIX** - 고정 차트 스케일 모드.

```
//+------------------------------------------------------------------+
//| 고정 축척 모드가 활성화되어 있는지 확인합니다                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정 스케일 모드를 활성화/비활성화합니다                                  |
//+------------------------------------------------------------------+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SCALEFIX_11** - 1:1 차트 스케일 모드.

```
//+------------------------------------------------------------------+
//| 1:1 배율이 사용 가능한지 확인합니다                                    |
//+------------------------------------------------------------------+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| "1:1" 스케일 모드를 활성화/비활성화                                |
//+------------------------------------------------------------------+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SCALE_PT_PER_BAR** - 막대당 포인트 단위로 차트 축척을 지정하는 모드.

```
//+------------------------------------------------------------------+
//| "막대당 포인트 수" 차트 축척 모드가 활성화되어 있는지 확인합니다     |
//+------------------------------------------------------------------+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
```

© 2000-2025, MetaQuotes Ltd.

```
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| "막대당 포인트" 차트 스케일링 모드를 활성화/비활성화합니다.          |
//+------------------------------------------------------------------+
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_OHLC** - 왼쪽 상단 모서리에 OHLC 값을 표시하는 속성.

```
//+------------------------------------------------------------------+
//| 차트의 왼쪽 상단 모서리에 OHLC 값 표시가 활성화되어 있는지 확인합니다 |
//+------------------------------------------------------------------+
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트의 왼쪽 상단 모서리에 OHLC 값을 표시하거나 표시하지 않습니다 |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_BID_LINE** - 차트에 입찰 값을 수평선으로 표시하는 속성.

```
//+------------------------------------------------------------------+
//| 차트에 입찰선 표시가 활성화되어 있는지 확인합니다                |
//+------------------------------------------------------------------+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에 입찰 선 표시를 활성화/비활성화합니다                      |
//+------------------------------------------------------------------+
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_ASK_LINE** - 차트에서 요청가를 수평선으로 표시하는 속성.

```
//+------------------------------------------------------------------+
//| 차트에 요청선 표시가 활성화되어 있는지 확인합니다                |
//+------------------------------------------------------------------+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 관리도에 요청선 표시를 활성화/비활성화합니다                     |
//+------------------------------------------------------------------+
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_SHOW_LAST_LINE** - 차트에 종가를 수평선으로 표시하는 속성.

```
//+------------------------------------------------------------------------------+
//| 마지막으로 수행된 거래 가격에 대한 라인 표시가 활성화되어 있는지 확인합니다 |
//+------------------------------------------------------------------------------+
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------------------+
//| 마지막으로 수행된 거래 가격에 대한 라인 표시를 활성화/비활성화합니다 |
//+------------------------------------------------------------------------------+
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_PERIOD_SEP** - 인접 기간 간의 수직 구분 표시 특성.

```
//+------------------------------------------------------------------------------+
//| 인접 기간 간의 수직 구분 기호 표시가 활성화되어 있는지 확인합니다 |
//+------------------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------------------+
//| 인접 기간 간의 수직 구분 기호 표시 활성화/비활성화 |
//+------------------------------------------------------------------------------+
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_GRID** - 차트 그리드 표시 속성.

```
//+------------------------------------------------------------------+
//| 차트 그리드가 표시되는지 확인합니다                              |
//+------------------------------------------------------------------+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에 그리드 표시를 활성화/비활성화합니다                       |
//+------------------------------------------------------------------+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_VOLUMES** - 도표에 볼륨을 표시하는 속성.

```
//+------------------------------------------------------------------+
//| 볼륨이 차트에 표시되는지 확인합니다                              |
//| 플래그는 모드를 표시하는 볼륨을 나타냅니다                       |
//+------------------------------------------------------------------+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=WRONG_VALUE;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
```

© 2000-2025, MetaQuotes Ltd.

```
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+------------------------------------------------------------------+
//| 차트에 볼륨을 표시하는 모드를 설정합니다                                  |
//+------------------------------------------------------------------+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_OBJECT_DESCR** - 그래픽 객체 팝업 설명의 속성.

```
//+------------------------------------------------------------------+
//| 마우스를 그래픽 개체 위에 놓을 때 그래픽 개체의  |
//| 팝업 설명이 표시되는지 확인합니다                                  |
//+------------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 그래픽 객체의 팝업 설명 표시 활성화/비활성화 |
```

© 2000-2025, MetaQuotes Ltd.

```
//| 마우스가 그것들을 호버할 때                                                    |
//+------------------------------------------------------------------------------+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_VISIBLE_BARS**는 차트에서 표시할 수 있는 막대 수를 정의합니다.

```
//+------------------------------------------------------------------------------+
//| 차트 창에 표시되는(표시되는) 막대 수를 가져옵니다 |
//+------------------------------------------------------------------------------+
int ChartVisibleBars(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
```

- **CHART_WINDOWS_TOTAL**는 지표 하위 창을 포함하여 차트 창의 총 수를 정의합니다.

```
//+------------------------------------------------------------------------------+
//| 지표 하위 창을 포함한 전체 차트 창 수를 가져옵니다 |
//+------------------------------------------------------------------------------+
int ChartWindowsTotal(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
```

- **CHART_WINDOW_IS_VISIBLE**는 하위 창의 가시성을 정의합니다.

```
//+------------------------------------------------------------------+
//| 현재 차트 창 또는 하위 창이 표시되는지 확인합니다       |
//+------------------------------------------------------------------+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
```

- **CHART_WINDOW_HANDLE**은 차트 핸들을 반환합니다.

```
//+------------------------------------------------------------------+
//| 차트 핸들 가져오기                                               |
//+------------------------------------------------------------------+
int ChartWindowsHandle(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
//--- 오류 값 리셋
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
```

- **CHART_WINDOW_YDISTANCE**는 지표 하위 창의 위쪽 프레임과 차트의 기본 창의 위쪽 프레임 사이의 거리를 픽셀 단위로 정의합니다.

```
//+------------------------------------------------------------------+
//| 차트의 기본 창의 위쪽 경계와 하위 창의           |
//| 위쪽 경계 사이의 거리(픽셀)를 가져옵니다             |
//+------------------------------------------------------------------+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
```

- **CHART_FIRST_VISIBLE_BAR**는 차트에 표시되는 첫 번째 막대의 수를 반환합니다(막대 인덱싱은 시계열에 해당).

```
//+------------------------------------------------------------------------+
//| 차트에 표시되는 첫 번째 막대의 인덱스를 가져옵니다.                    |
//| 인덱싱은 시계열에서처럼 수행되며, 최신 막대의 인덱스가 가장 작습니다. |
//+------------------------------------------------------------------------+
int ChartFirstVisibleBar(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
```

- **CHART_WIDTH_IN_BARS**는 차트 너비를 막대 단위로 반환합니다.

```
//+------------------------------------------------------------------+
//| 차트의 너비(막대 단위)를 가져옵니다                                |
//+------------------------------------------------------------------+
int ChartWidthInBars(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
```

- **CHART_WIDTH_IN_PIXELS** 차트 너비(픽셀)를 반환합니다.

```
//+------------------------------------------------------------------+
//| 차트 너비(픽셀) 가져오기                                          |
//+------------------------------------------------------------------+
int ChartWidthInPixels(const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
```

- **CHART_HEIGHT_IN_PIXELS** - 차트 높이 속성(픽셀).

```
//+------------------------------------------------------------------+
//| 차트의 높이(픽셀)를 가져오기                                     |
//+------------------------------------------------------------------+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long result=-1;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((int)result);
  }
//+------------------------------------------------------------------+
//| 차트의 높이를 설정합니다(픽셀 단위)                              |
//+------------------------------------------------------------------+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_windo
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_BACKGROUND** - 차트 배경색.

```
//+------------------------------------------------------------------+
//| 차트 배경색 가져오기                                             |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
color ChartBackColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 차트 배경색 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 차트 배경색 설정                                                 |
//+------------------------------------------------------------------+
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 차트 배경색을 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_FOREGROUND** - 축, 스케일, OHLC 선의 색상.

```
//+------------------------------------------------------------------+
//| 축, 스케일, OHLC 선의 색상 가져오기                              |
//+------------------------------------------------------------------+
color ChartForeColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 축, 스케일, OHLC 선의 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 축, 스케일, OHLC 선 색상 설정                                    |
//+------------------------------------------------------------------+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 축, 스케일 및 OHLC 선 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_GRID** - chart grid color.

```
//+------------------------------------------------------------------+
//| 차트 그리드 색상 가져오기                                        |
//+------------------------------------------------------------------+
color ChartGridColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 차트 그리드 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 차트 그리드 색상 설정                                            |
//+------------------------------------------------------------------+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 값 리셋
   ResetLastError();
//--- 차트 그리드 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_VOLUME** – 볼륨 및 포지션 오프닝 레벨의 색상.

```
//+------------------------------------------------------------------+
//| 볼륨 및 마켓 엔트리 레벨의 색상 가져오기                         |
//+------------------------------------------------------------------+
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 볼륨 및 마켓 엔트리 레벨 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 볼륨 및 시장 엔트리 레벨의 색상 설정                             |
//+------------------------------------------------------------------+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 볼륨 및 시장 엔트리 레벨 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

- **CHART_COLOR_CHART_UP** - 위쪽 막대의 색깔, 막대 모양의 그림자와 막대기 몸통 테두리.

```
//+------------------------------------------------------------------------------+
//| 막대 모양의 위쪽, 그림자 및 막대 모양의 막대 모양 막대 모양 가져오기 |
//+------------------------------------------------------------------------------+
color ChartUpColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 막대의 빛깔과 그 그림자와 촛대의 테두리를 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 촛대 본체의 위쪽 막대, 그림자 및 테두리의 색상을 설정 |
//+------------------------------------------------------------------+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 쇠막대기의 색상과 그 그림자와 바디의 테두리 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_CHART_DOWN** - 다운 막대, 그 음영, 약세 캔들스틱 테두리의 색상.

```
//+------------------------------------------------------------------------------+
//| 다운 막대, 그 음영, 약세 캔들스틱 테두리의 색상 가져오기 |
//+------------------------------------------------------------------------------+
color ChartDownColorGet(const long chart_ID=0)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 다운 막대, 그 음영, 약세 캔들스틱 테두리 색상 수려
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------------------+
//| 약세 캔들스틱의 다운 막대, 그림자 및 테두리의 색상을 설정합니다 |
//+------------------------------------------------------------------------------+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 약세 촛대 몸통의 아래쪽 막대의 색깔과 그림자와 테두리 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_CHART_LINE** - 차트 선 및 도지 촛대의 색상.

```
//+------------------------------------------------------------------+
//| 차트 선과 도지 촛대의 색상 가져오기                |
//+------------------------------------------------------------------+
color ChartLineColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 차트 선과 도지 촛대의 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 차트 선과 도지 촛대의 색상 설정                                  |
//+------------------------------------------------------------------+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 차트 선과 도지 촛대의 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BULL** - 상승 촛대의 바디 색깔.

```
//+------------------------------------------------------------------+
//| 상승 촛대 바디 색상을 가져오기                                   |
//+------------------------------------------------------------------+
color ChartBullColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 상승 캔들스틱 바디 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 상승세 캔들스틱 바디 색상 설정                                   |
//+------------------------------------------------------------------+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- 상승세 캔들스틱 바디 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BEAR** - 상승세 캔들스틱 바디 색상.

```
//+------------------------------------------------------------------+
//| 상승세 캔들스틱 바디 색상 가져오기                                |
//+------------------------------------------------------------------+
color ChartBearColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 상승세 캔들스틱 바디 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 상승세 캔들스틱 바디 색상 설정                                   |
//+------------------------------------------------------------------+
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 상승세 캔들스틱 바디 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
```

- **CHART_COLOR_BID** - 요청가 선 색상.

```
//+------------------------------------------------------------------+
//| 요청 선 색상 가져오기                                            |
//+------------------------------------------------------------------+
color ChartBidColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 요청가 라인 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 입찰 선 색상 설정                                                |
//+------------------------------------------------------------------+
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 입찰가 선 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_ASK** - 요청가 선 색상.

```
//+------------------------------------------------------------------+
//| 요청 선 색상 가져오기                                            |
//+------------------------------------------------------------------+
color ChartAskColorGet(const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 요청가 라인 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 요청 선 색상 설정                                                |
//+------------------------------------------------------------------+
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 요청가 라인 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_LAST** - 최근 수행한 거래 가격 선 (종가) 색상.

```
//+------------------------------------------------------------------+
//| 최근 수행 거래 가격 선 색상 가져오기                             |
//+------------------------------------------------------------------+
color ChartLastColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 최근 수행 거래 가격선(종가) 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 최근 수행 거래 가격선 색상 설정                                  |
//+------------------------------------------------------------------+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 최근 수행 거래가 라인(종가) 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_COLOR_STOP_LEVEL** - 스탑 오더 레벨 색상(스탑 로스 및 이익 실현).

```
//+------------------------------------------------------------------+
//| 스탑 로스 및 이익 실현 색상 가져오기                             |
//+------------------------------------------------------------------+
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- 변수를 준비하여 색을 입힘
   long result=clrNONE;
//--- 오류 값 리셋
   ResetLastError();
//--- 스탑 오더 수준 (스탑 로스 및 이익 실현) 색상 수령
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return((color)result);
  }
//+------------------------------------------------------------------+
//| 스탑 로스 및 이익 실현 수준 색상 설정                            |
//+------------------------------------------------------------------+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 스탑 오더 수준 (스탑 로스 및 이익 실현) 색상 설정
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** - 차트에 거래 수준을 표시하는 속성(오픈 포지션 수준, 스탑 로스, 이익 실현 및 보류 중 주문)

```
//+------------------------------------------------------------------+
//| 거래 수준이 차트에 표시되는지 확인합니다                         |
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 거래 수준 표시 활성화/비활성화                                   |
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** - 마우스를 사용하여 차트에서 거래 수준을 드래그하는 기능을 활성화하는 속성.

```
//+------------------------------------------------------------------+
//| 마우스를 사용하여 차트에서 거래 수준을 드래그할 수 있는지 확인 |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 마우스를 사용하여 차트에서 거래 수준 드래그 활성화/비활성화 |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

- **CHART_SHOW_DATE_SCALE** - 차트에 시간 스케일을 표시하는 속성.

```
//+------------------------------------------------------------------+
//| 시간 스케일이 차트에 표시되는지 확인합니다                       |
//+------------------------------------------------------------------+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에 시간 스케일 표시 활성화/비활성화                          |
//+------------------------------------------------------------------+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_PRICE_SCALE** - 차트에 가격 스케일을 표시하는 속성.

```
//+------------------------------------------------------------------+
//| 가격 스케일이 차트에 표시되는지 확인                             |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에 가격 척도 표시 활성화/비활성화                            |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHOW_ONE_CLICK** - 차트에 "원클릭 트레이딩" 패널 표시 속성.

```
//+------------------------------------------------------------------+
//| 차트에 "원클릭 트레이딩" 패널이 표시되는지 확인                  |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 가져오기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 차트 속성 값을 메모리에 저장합니다
   result=value;
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 원클릭 트레이딩 패널 표시 활성화/비활성화      |
//| 차트에서                                                         |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_SHIFT_SIZE** - 오른쪽 테두리에서 0 막대의 이동 크기(백분율 값).

```
//+------------------------------------------------------------------+
//| 차트의 오른쪽 테두리에서 0 막대의 이동 크기를 백분율 값 |
//| (10% ~ 50%)으로 가져옵니다           |
//+------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- 결과를 얻기 위해 변수를 준비
   double result=EMPTY_VALUE;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return(result);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------------------+
//| 차트의 오른쪽 테두리에서 0 막대의 이동 크기를                                |
//| 백분율 값(10% ~ 50%)으로 가져옵니다.                                         |
//| Shift 모드를 활성화하려면 CHART_SHIFT 속성 값을 true로 설정해야 합니다. |
//+------------------------------------------------------------------------------+
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_FIXED_POSITION** - 차트 고정 위치를 왼쪽 테두리부터 백분율 값으로 표시.

```
//+------------------------------------------------------------------------------+
//| 왼쪽 테두리(백분율 값)에서 차트의 고정 위치 위치를 가져옵니다 |
//+------------------------------------------------------------------------------+
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- 결과를 얻기 위해 변수를 준비
   double result=EMPTY_VALUE;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return(result);
  }
//+------------------------------------------------------------------------------+
//| 왼쪽 테두리(백분율 값)에서 차트의 고정 위치 위치를 가져옵니다. |
//| 차트의 고정 위치 위치를 보려면 CHART_AUTOSCROLL 속성 값을  |
//| false로 설정해야 합니다.
//+------------------------------------------------------------------------------+
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_FIXED_MAX** - 차트의 고정 최대값 속성.

```
//+------------------------------------------------------------------+
//| 차트의 고정 최대값 값을 가져옵니다                                |
//+------------------------------------------------------------------+
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- 결과를 얻기 위해 변수를 준비
   double result=EMPTY_VALUE;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return(result);
  }
//+------------------------------------------------------------------+
//| 차트의 고정 최대값을 설정합니다.                                  |
//| 속성 값을 변경하려면 CHART_SCALEFIX 속성 값을                    |
//| 임시로 true로 설정해야 합니다.                                    |
//+------------------------------------------------------------------+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
    }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_FIXED_MIN** - 차트의 고정 최소값 속성.

```
//+------------------------------------------------------------------+
//| 차트의 고정 최소값 속성                                          |
//+------------------------------------------------------------------+
double ChartFixedMinGet(const long chart_ID=0)
  {
//--- 결과를 얻기 위해 변수를 준비
   double result=EMPTY_VALUE;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return(result);
  }
//+------------------------------------------------------------------+
//| 차트의 고정 최소값을 설정합니다.                                 |
//| 속성 값을 변경하려면 CHART_SCALEFIX 속성 값을                   |
//| 임시로 true로 설정해야 합니다.                                   |
//+------------------------------------------------------------------+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_POINTS_PER_BAR** - 막대당 포인트 단위의 척도 값.

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 막대당 점 단위의 차트 축척 값을 가져옵니다                       |
//+------------------------------------------------------------------+
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- 결과를 얻기 위해 변수를 준비
   double result=EMPTY_VALUE;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return(result);
  }
//+------------------------------------------------------------------+
//| 막대당 포인트 단위로 차트 스케일 값을 설정합니다.                |
//| 이 속성의 값 변경 결과를 보려면      |
//| CHART_SCALE_PT_PER_BAR 속성 값을 true로 미리 설정해야 합니다. |
//+------------------------------------------------------------------+
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_PRICE_MIN**는 차트 최소값을 반환합니다.

```
//+------------------------------------------------------------------+
//| 기본 창 또는 하위 창에서 차트 최소값을 가져옵니다 |
//+------------------------------------------------------------------+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- 결과를 얻기 위해 변수를 준비
   double result=EMPTY_VALUE;
//--- 오류 값 리셋
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return(result);
  }
```

- **CHART_PRICE_MAX**는 차트 최대값을 반환합니다.

```
//+------------------------------------------------------------------+
//| 기본 창 또는 하위 창에서 차트 최대값을 가져옵니다 |
//+------------------------------------------------------------------+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
//--- 결과를 얻기 위해 변수를 준비
   double result=EMPTY_VALUE;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
     }
//--- 차트 속성 값을 반환합니다
   return(result);
  }
```

- **CHART_COMMENT** - 차트 내 주석.

```
//+------------------------------------------------------------------+
//| 차트의 왼쪽 상단 모서리에 주석을 가져옵니다                      |
//+------------------------------------------------------------------+
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 받기
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트의 왼쪽 상단 모서리에 주석을 가져옵니다                      |
//+------------------------------------------------------------------+
bool ChartCommentSet(const string str,const long chart_ID=0)
  {
//--- 오류 값 리셋
   ResetLastError();
//--- 속성값 설정
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //--- 전문가 저널에 오류 메시지를 표시합니다
      Print(__FUNCTION__+", 에러 코드 = ",GetLastError());
      return(false);
     }
//--- 성공적 실행
   return(true);
  }
```

- **CHART_IS_MAXIMIZED** - 차트창이 최대화됩니다.

```
//+------------------------------------------------------------------+
//| 현재 차트 창을 최대화할지 여부를 정의합니다                      |
//+------------------------------------------------------------------+
bool ChartWindowsIsMaximized(bool &result,const long chart_ID=0)
  {
//--- 속성 값을 받기 위해 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 수령
   if(!ChartGetInteger(chart_ID,CHART_IS_MAXIMIZED))
     {
      //--- Experts 로그에 오류 메시지 표시
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 변수 내 차트 속성값 저장
   result=value;
//--- 실행 성공
   return(true);
  }
```

- **CHART_IS_MINIMIZED** - 차트창이 최소화됩니다.

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 현재 차트 창을 최소화할지 여부를 정의합니다                      |
//+------------------------------------------------------------------+
bool ChartWindowsIsMinimized(bool &result,const long chart_ID=0)
  {
//--- 속성 값 수령을 위한 변수 준비
   long value;
//--- 오류 값 리셋
   ResetLastError();
//--- 속성 값 수령
   if(!ChartGetInteger(chart_ID,CHART_IS_MINIMIZED))
     {
      //--- Experts 로그에 오류 메시지 표시
      Print(__FUNCTION__+", Error Code = ",GetLastError());
      return(false);
     }
//--- 변수 내 차트 속성 값 저장
   result=value;
//--- 실행 성공
   return(true);
  }
```

차트 속성 패널

```
//--- 제어 요소 라이브러리를 연결
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- 미리 정의된 상수
#define X_PROPERTY_NAME_1    10  // 첫 번째 열에 있는 특성 이름의 x 좌표
#define X_PROPERTY_VALUE_1   225 // 첫 번째 열에 있는 특성 값의 x 좌표
#define X_PROPERTY_NAME_2    345 // 두 번째 및 세 번째 열에 있는 특성 이름의 x 좌표
#define X_PROPERTY_VALUE_2   550 // 두 번째 및 세 번째 열에 있는 특성 값의 x 좌표
#define X_BUTTON_1           285 // 첫 번째 열에 있는 버튼의 x 좌표
#define X_BUTTON_2           700 // 두 번째 열에 있는 버튼의 x 좌표
#define Y_PROPERTY_1         30  // 첫 번째 및 두 번째 열의 시작의 y 좌표
#define Y_PROPERTY_2         286 // 세 번째 열의 시작의 y 좌표
#define Y_DISTANCE           16  // 선 사이의 y축 간격
#define LAST_PROPERTY_NUMBER 111 // 마지막 그래픽 특성 수
//--- 입력 매개 변수
input color InpFirstColor=clrDodgerBlue; // 홀수 선 색상
input color InpSecondColor=clrGoldenrod; // 짝수 선 색상
//--- 변수 및 배열
CChartObjectLabel  ExtLabelsName[];  // 특성 이름 표시 레이블
CChartObjectLabel  ExtLabelsValue[]; // 특성 값 표시 레이블
CChartObjectButton ExtButtons[];     // 버튼
int                ExtNumbers[];     // 속성 지수
string             ExtNames[];       // 속성명
uchar              ExtDataTypes[];   // 속성 데이터 유형 (integer, double, string)
uint               ExtGroupTypes[];  // 그룹 중 하나에 속성을 속성에 데이터를 저장하는 배열
uchar              ExtDrawTypes[];   // 속성 표시 유형에 데이터를 저장하는 배열
double             ExtMaxValue[];    // 패널 작업 시 가능한 최대 속성 값
double             ExtMinValue[];    // 패널 작업 시 가능한 최소 속성 값
```

© 2000-2025, MetaQuotes Ltd.

```
double              ExtStep[];        // 속성 변경 단계
int                 ExtCount;         // 모든 속성의 총 수
color               ExtColors[2];     // 라인 표시용 색상 배열
string              ExtComments[2];   // 주석 배열 (CHART_COMMENT 속성용)
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 차트에 주석 표시
   Comment("SomeComment");
//--- 나중에 전환할 수 있도록 배열에 색상 저장
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- 나중에 주석을 전환할 수 있도록 배열에 주석 저장
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- 차트 속성 관리를 위한 제어판 준비 및 표시
   if(!PrepareControls())
      return(INIT_FAILED);
//--- 성공적 실행
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| expert 함수 초기화해제                                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 차트 내 주석 삭제
   Comment("");
  }
//+------------------------------------------------------------------+
//| 차트 이벤트 핸들러                                               |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- 차트 객체 클릭 이벤트 확인
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- 개체명을 구분 기호로 나눔
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
      //--- 객체가 버튼인지 체크
      if(obj_name[0]=="Button")
        {
         //--- 버튼 인덱스 수령
```

© 2000-2025, MetaQuotes Ltd.

```
         int index=(int)StringToInteger(obj_name[1]);
         //--- 버튼 누름해제
         ExtButtons[index].State(false);
         //--- 속성의 유형에 따라 새 값을 설정
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- 속성 값 다시 그리기
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| 차트 정수 속성 변경                                              |
//+------------------------------------------------------------------+
void ChangeIntegerProperty(const int index)
  {
//--- 현재 속성 값 수령
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- 다음 속성 값 정의
   switch(ExtDrawTypes[index])
     {
      case 'C':
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- 새 속성 값 설정
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------+
//| 차트 doubl e 속성 변경                                           |
//+------------------------------------------------------------------+
void ChangeDoubleProperty(const int index)
  {
//--- 현재 속성 값 수령
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- 다음 속성 값 정의
   value=GetNextValue(value,index);
//--- 새 속성 값 설정
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 차트 문자열 속성 변경                                            |
//+------------------------------------------------------------------+
void ChangeStringProperty(const int index)
  {
//--- ExtComments 배열 내에서 전환하기 위한 정적 변수
   static uint comment_index=1;
//--- 다른 주석을 수신하기 위한 색인 변경
   comment_index=1-comment_index;
//--- 새 속성 값 설정
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
  }
//+------------------------------------------------------------------+
//| 다음 특성 값을 가져옵니다                                        |
//+------------------------------------------------------------------+
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+------------------------------------------------------------------+
//| 색상 유형 속성에 대한 다음 색상을 가져옵니다                     |
//+------------------------------------------------------------------+
color GetNextColor(const color clr)
  {
//--- 다음 색상 값 반환
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+------------------------------------------------------------------+
//| 속성 값 다시 그리기                                              |
//+------------------------------------------------------------------+
void RedrawProperties(void)
  {
//--- 속성 값 텍스트
   string text;
   long   value;
//--- 속성 수 루프
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          case 'I':
             //--- 현재 속성 값 수령
             if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
             break;
             //--- 정수 속성 텍스트
             switch(ExtDrawTypes[i])
               {
                //--- 색상 속성
                case 'C':
                   text=(string)((color)value);
                   break;
                   //--- boolean 속성
                case 'B':
                   text=(string)((bool)value);
                   break;
                   //--- ENUM_CHART_MODE 열거 속성
                case 'M':
                   text=EnumToString((ENUM_CHART_MODE)value);
                   break;
                   //--- ENUM_CHART_VOLUME_MODE 열거 속성
                case 'V':
                   text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                   break;
                   //--- int 타입 수
                default:
                   text=IntegerToString(value);
                   break;
               }
             break;
          case 'D':
             //--- double 속성 텍스트
             text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumber
             break;
          case 'S':
             //--- 문자열 속성 텍스트
             text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
             break;
         }
       //--- 속성 값 표시
       ExtLabelsValue[i].Description(text);
     }
   }
//+------------------------------------------------------------------+
//| 차트 속성 관리용 패널 생성                                       |
//+------------------------------------------------------------------+
bool PrepareControls(void)
   {
//--- 예약된 배열에 대해 메모리 할당
```

© 2000-2025, MetaQuotes Ltd.

```
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- 변수
   int i=0;     // 루프 변수
   int col_1=0; // 첫 칼럼에의 속성 수
   int col_2=0; // 둘째 칼럼의 속성 수
   int col_3=0; // 셋째 칼럼의 속성 수
//--- 속성의 현재 수 - 0
   ExtCount=0;
//--- 루프에 있는 속성 찾기
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- 속성의 현재 수 저장
      ExtNumbers[ExtCount]=i;
      //--- 루프 변수 값 증가
      i++;
      //--- 속성에 그러한 수가 있는지 확인
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],Ex
        {
         //--- 속성용 제어 요소 생성
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- 속성용 레이블 및 버튼 생성
               if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTT
               return(false);
               //--- 첫 번째 열의 요소 수가 증가했습니다
               col_1++;
               break;
            case 2:
               //--- 속성용 레이블 및 버튼 생성
               if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTT
               return(false);
               //--- 두 번째 열의 요소 수가 증가했습니다
               col_2++;
               break;
            case 3:
               //--- 속성에 대한 레이블만 작성합니다
               if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_PR
               return(false);
               //--- 세 번째 열의 요소 수가 증가했습니다
               col_3++;
               break;
           }
         //--- 최대 및 최소 특성 값 및 단계 정의
         GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount
         //--- 속성 수 증가
         ExtCount++;
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 배열에서 사용하지 않는 메모리 사용 가능
   MemoryAllocation(ExtCount);
//--- 속성 값 다시 그리기
   RedrawProperties();
   ChartRedraw();
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 배열용 메모리 할당                                               |
//+------------------------------------------------------------------+
void MemoryAllocation(const int size)
  {
   ArrayResize(ExtLabelsName,size);
   ArrayResize(ExtLabelsValue,size);
   ArrayResize(ExtButtons,size);
   ArrayResize(ExtNumbers,size);
   ArrayResize(ExtNames,size);
   ArrayResize(ExtDataTypes,size);
   ArrayResize(ExtGroupTypes,size);
   ArrayResize(ExtDrawTypes,size);
   ArrayResize(ExtMaxValue,size);
   ArrayResize(ExtMinValue,size);
   ArrayResize(ExtStep,size);
  }
//+------------------------------------------------------------------+
//| 속성 인덱스가 다음 중 하나에 속하는지 확인합니다                 |
//| ENUM_CHART_PROPERTIES 열거값                                     |
//+------------------------------------------------------------------+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &d
  {
//--- 속성이 정수 유형인지 확인합니다
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                        // ENUM_CHART_PROPERTY_INTEGER 열거값 속성
      GetTypes(ind,group_type,draw_type); // 특성 표시 매개 변수 정의
      return(true);
     }
//--- 속성이 double 타입인지 확인
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
      data_type='D';                        // ENUM_CHART_PROPERTY_DOUBLE 열거값 속성
      GetTypes(ind,group_type,draw_type); // 특성 표시 매개 변수 정의
      return(true);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 속성이 문자열 유형인지 확인
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                        // ENUM_CHART_PROPERTY_STRING 열거값 속성
      GetTypes(ind,group_type,draw_type); // 특성 표시 매개 변수 정의
      return(true);
     }
//--- 속성이 어떤 열거형에도 속하지 않습니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 속성을 저장할 그룹을 정의합니다                                  |
//| 그것으 표시 유형과 마찬가지로                                    |
//+------------------------------------------------------------------+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- 속성이 세 번째 그룹에 속하는지 확인합니다
//--- 세 번째 그룹 속성이 CHART_BRING_TO_TOP 에서 시작하는 두 번째 열에 표시됩니다
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- 속성이 두 번째 그룹에 속하는지 확인합니다
//--- 두 번째 그룹 속성이 두 번째 열의 시작 부분에 표시됩니다
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//--- 여기서 잘 보면 속성이 첫 번째 그룹(첫 번째 열)에 속합니다
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+------------------------------------------------------------------+
//| 속성이 세 번째 그룹에 속하는지 확인하고                          |
//| 긍정 응답인 경우 해당 표시 유형을 정의합니다                     |
//+------------------------------------------------------------------+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- 속성이 세 번째 그룹에 속하는지 확인합니다
   switch(property_number)
     {
      //--- boolean 속성
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- integer 속성
      case CHART_VISIBLE_BARS:
      case CHART_WINDOWS_TOTAL:
      case CHART_WINDOW_HANDLE:
      case CHART_WINDOW_YDISTANCE:
      case CHART_FIRST_VISIBLE_BAR:
```

© 2000-2025, MetaQuotes Ltd.

```
      case CHART_WIDTH_IN_BARS:
      case CHART_WIDTH_IN_PIXELS:
         draw_type='I';
         break;
         //--- double 속성
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- 사실 이 속성은 차트를 다른 모든 차트 위에 표시하는 명령입니다
         //--- 창이 항상 열려 있으므로 이 패널을 적용할 필요가 없습니다
         //--- 우리가 사용하기 전에 다른 것들 위에
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- 속성이 세 번째 그룹에 속하지 않습니다
      default:
         return(false);
     }
//--- 속성이 세 번째 그룹에 속하지 않습니다
   group_type=3;
   return(true);
  }
//+------------------------------------------------------------------+
//| 속성이 두 번째 그룹에 속하는지 확인하고                          |
//| 긍정 응답인 경우 해당 표시 유형을 정의합니다                     |
//+------------------------------------------------------------------+
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- 속성이 두 번째 그룹에 속하는지 확인합니다
   switch(property_number)
     {
      //--- ENUM_CHART_MODE 유형 속성
      case CHART_MODE:
         draw_type='M';
         break;
         //--- ENUM_CHART_VOLUME_MODE 유형 속성
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- string 속성
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- 색 속성
      case CHART_COLOR_BACKGROUND:
      case CHART_COLOR_FOREGROUND:
      case CHART_COLOR_GRID:
      case CHART_COLOR_VOLUME:
```

© 2000-2025, MetaQuotes Ltd.

```
      case CHART_COLOR_CHART_UP:
      case CHART_COLOR_CHART_DOWN:
      case CHART_COLOR_CHART_LINE:
      case CHART_COLOR_CANDLE_BULL:
      case CHART_COLOR_CANDLE_BEAR:
      case CHART_COLOR_BID:
      case CHART_COLOR_ASK:
      case CHART_COLOR_LAST:
      case CHART_COLOR_STOP_LEVEL:
         draw_type='C';
         break;
         //--- 속성이 두 번째 그룹에 속하지 않습니다
      default:
         return(false);
     }
//--- 속성이 두 번째 그룹에 속합니다
   group_type=2;
   return(true);
  }
//+------------------------------------------------------------------+
//| 속성이 두 번째 및 세 번째 속성 그룹에 속하지 않는 것으로      |
//| 이미 알려진 경우에만 호출됩니다                                  |
//+------------------------------------------------------------------+
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- 속성이 첫 번째 그룹에 속합니다
   group_type=1;
//--- 속성 표시 유형 정의
   switch(property_number)
     {
      //--- integer 속성
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- double 속성
      case CHART_SHIFT_SIZE:
      case CHART_FIXED_POSITION:
      case CHART_FIXED_MAX:
      case CHART_FIXED_MIN:
      case CHART_POINTS_PER_BAR:
         draw_type='D';
         return;
         //--- 부울 속성만 남아 있습니다
      default:
         draw_type='B';
         return;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 속성용 레이블 및 버튼 생성                                       |
//+------------------------------------------------------------------+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
  {
//--- ExtColors 색상 배열 내에서 전환하기 위한 정적 배열
   static uint color_index[3]={1,1,1};
//--- 다른 색상 수신에 대한 인덱스 변경
   color_index[type]=1-color_index[type];
//--- 속성에 대한 표시 레이블 및 버튼(btn=true인 경우)
   if(!LabelCreate(ExtLabelsName[ind],"name_"+(string)ind,ExtNames[ind],ExtColors[colo
      return(false);
   if(!LabelCreate(ExtLabelsValue[ind],"value_"+(string)ind,"",ExtColors[color_index[t
      return(false);
   if(btn && !ButtonCreate(ExtButtons[ind],(string)ind,xb,y+1))
      return(false);
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 레이블 생성                                                      |
//+------------------------------------------------------------------+
bool LabelCreate(CChartObjectLabel &lbl,const string name,const string text,
                 const color clr,const int x,const int y)
  {
   if(!lbl.Create(0,"Label_"+name,0,x,y)) return(false);
   if(!lbl.Description(text))             return(false);
   if(!lbl.FontSize(10))                  return(false);
   if(!lbl.Color(clr))                    return(false);
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
//| 버튼 생성                                                        |
//+------------------------------------------------------------------+
bool ButtonCreate(CChartObjectButton &btn,const string name,
                  const int x,const int y)
  {
   if(!btn.Create(0,"Button_"+name,0,x,y,50,15)) return(false);
   if(!btn.Description("Next"))                  return(false);
   if(!btn.FontSize(10))                         return(false);
   if(!btn.Color(clrBlack))                      return(false);
   if(!btn.BackColor(clrWhite))                  return(false);
   if(!btn.BorderColor(clrBlack))                return(false);
//--- 성공적 실행
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 최대 및 최소 속성 값과 단계를 정의합니다                          |
//+------------------------------------------------------------------+
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- 속성 유형에 따라 값 설정
   switch(property_number)
     {
      case CHART_SCALE:
         max=5;
         min=0;
         step=1;
         break;
      case CHART_MODE:
      case CHART_SHOW_VOLUMES:
         max=2;
         min=0;
         step=1;
         break;
      case CHART_SHIFT_SIZE:
         max=50;
         min=10;
         step=2.5;
         break;
      case CHART_FIXED_POSITION:
         max=90;
         min=0;
         step=15;
         break;
      case CHART_POINTS_PER_BAR:
         max=19;
         min=1;
         step=3;
         break;
      case CHART_FIXED_MAX:
         value=ChartGetDouble(0,CHART_FIXED_MAX);
         max=value*1.25;
         min=value;
         step=value/32;
         break;
      case CHART_FIXED_MIN:
         value=ChartGetDouble(0,CHART_FIXED_MIN);
         max=value;
         min=value*0.75;
         step=value/32;
         break;
      case CHART_HEIGHT_IN_PIXELS:
         max=700;
         min=520;
```

© 2000-2025, MetaQuotes Ltd.

```
         step=30;
         break;
         //--- 디폴트 값
      default:
         max=1;
         min=0;
         step=1;
   }
}
```

© 2000-2025, MetaQuotes Ltd.

개체 상수

44개의 그래픽 개체를 만들어 가격 차트에 표시할 수 있습니다. 개체 작업에 대한 모든 상수는 다음과 같이 9개 그룹으로 나뉩니다:

- 개체 유형 – 그래픽 개체의 식별자;
- 개체 속성 – 그래픽 개체의 특성 설정 및 가져오기;
- 개체 바인딩 방법 – 차트에서 개체 위치 지정 상수;
- 바인딩 모서리 – 차트에서 개체가 위치한 모서리를 상대적으로 설정합니다;
- 개체의 가시성 – 개체가 보이는 시간 범위 설정;
- 엘리엇 파동 레벨 – 웨이브 그라데이션;
- 갠 개체 – 갠 팬 및 갠 그리드에 대한 추세 상수;
- 웹 색상 – 미리 정의된 웹 색상 상수;
- 윙딩 – 윙딩(Wingdings) 글꼴의 문자 코드.

© 2000-2025, MetaQuotes Ltd.

개체 유형

ObjectCreate() 함수를 사용하여 그래픽 만드는 경우 생성할 개체 유형을 지정해야 합니다. 이 유형은 ENUM_OBJECT 열거값 중 하나일 수 있습니다. 다음 작업을 위한 함수를 사용하여 개체 속성의 추가 사양을 수행할 수 있습니다: 그래픽 개체.

ENUM_OBJECT

ID		설명
OBJ_VLINE		수직선
OBJ_HLINE		수평선
OBJ_TREND		추세선
OBJ_TRENDBYANGLE		각도별 추세선
OBJ_CYCLES		주기선
OBJ_ARROWED_LINE		화살표 선
OBJ_CHANNEL		등거리 채널
OBJ_STDDEVCHANNEL		표준 편차 채널
OBJ_REGRESSION		선형 회귀 채널
OBJ_PITCHFORK		앤드류스 피치포크
OBJ_GANNLINE		갠(Gann) 선
OBJ_GANNFAN		갠(Gann) 팬
OBJ_GANNGRID		갠(Gann) 그리드
OBJ_FIBO		피보나치 되돌림
OBJ_FIBOTIMES		피보나치 타임존
OBJ_FIBOFAN		피보나치 팬
OBJ_FIBOARC		피보나치 아크
OBJ_FIBOCHANNEL		피보나치 채널
OBJ_EXPANSION		피보나치 확장
OBJ_ELLIOTWAVE5		엘리엇 동인 파동
OBJ_ELLIOTWAVE3		엘리엇 보정 파동
OBJ_RECTANGLE		직사각형
OBJ_TRIANGLE		삼각형
OBJ_ELLIPSE		타원
OBJ_ARROW_THUMB_UP		추천
OBJ_ARROW_THUMB_DOWN		비추천

© 2000-2025, MetaQuotes Ltd.

ID		설명
OBJ_ARROW_UP		위쪽 화살표
OBJ_ARROW_DOWN		아래쪽 화살표
OBJ_ARROW_STOP		중지 신호
OBJ_ARROW_CHECK		체크 기호
OBJ_ARROW_LEFT_PRICE		좌측 가격 레이블
OBJ_ARROW_RIGHT_PRICE		우측 가격 레이블
OBJ_ARROW_BUY		매수 기호
OBJ_ARROW_SELL		매도 기호
OBJ_ARROW		화살표
OBJ_TEXT		텍스트
OBJ_LABEL		레이블
OBJ_BUTTON		버튼
OBJ_CHART		차트
OBJ_BITMAP		비트맵
OBJ_BITMAP_LABEL		비트맵 레이블
OBJ_EDIT		편집
OBJ_EVENT		이코노믹 캘린더의 이벤트에 해당하는 "이벤트" 개체
OBJ_RECTANGLE_LABEL		사용자 지정 그래픽 인터페이스를 생성하고 설계하기 위한 "직사각형 레이블" 개체.

© 2000-2025, MetaQuotes Ltd.

OBJ_VLINE

수직선.

참조

수직선을 그릴 때 모든 차트 창에 대해 선 표시 모드(OBJPROP_RAY 속성)를 설정할 수 있습니다.

예

다음 스크립트는 차트에서 세로선을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩수직선₩ 그래픽 개체를 만들고 이동합니다."
#property description "고정점 날짜는 백분율로 설정됩니다."
#property description "차트 창 너비(막대 단위)."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="VLine";      // 선 이름
input int             InpDate=25;           // 이벤트 날짜(%)
input color           InpColor=clrRed;      // 선 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 선 스타일
input int             InpWidth=3;           // 선 너비
input bool            InpBack=false;        // 배경 선
input bool            InpSelection=true;    // 이동하려면 강조 표시
input bool            InpRay=true;          // 아래로 선 연장
input bool            InpHidden=true;       // 개체 목록에 숨겨짐
input long            InpZOrder=0;          // 마우스 클릭 우선 순위
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 수직선 생성                                                      |
//+------------------------------------------------------------------+
bool VLineCreate(const long            chart_ID=0,        // 차트의 ID
                 const string          name="VLine",      // 선 이름
                 const int             sub_window=0,      // 하위 창 인덱스
                 datetime              time=0,            // 선 시간
                 const color           clr=clrRed,        // 선 색상
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // 선 스타일
                 const int             width=1,           // 선 너비
                 const bool            back=false,        // 배경에
                 const bool            selection=true,    // 이동하려면 강조 표시
                 const bool            ray=true,          // 아래쪽으로 선 연장
                 const bool            hidden=true,       // 개체 목록에 숨겨짐
                 const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 선 시간이 설정되지 않은 경우 마지막 막대를 통해 그립니다
   if(!time)
      time=TimeCurrent();
//--- 오류 값 재설정
   ResetLastError();
//--- 수직선 생성
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": 수직선 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 선 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 표시 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 선을 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 차트 하위 창에 선 표시 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 수직선 이동                                                      |
//+------------------------------------------------------------------+
bool VLineMove(const long   chart_ID=0,   // 차트의 ID
               const string name="VLine", // 선 이름
               datetime     time=0)       // 선 시간
  {
//--- 선 시간이 설정되지 않은 경우 선을 마지막 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
//--- 오류 값 재설정
   ResetLastError();
//--- 수직선 이동
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": 수직선 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 수직선 실패                                                      |
//+------------------------------------------------------------------+
bool VLineDelete(const long   chart_ID=0,   // 차트의 ID
                 const string name="VLine") // 선 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 수직선 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 수직선 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("오류! 입력 매개변수의 잘못된 값!");
       return;
      }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 사용할 날짜 값을 저장하기 위한 배열
//--- 선 고정점 좌표 설정 및 변경
   datetime date[];
//--- 메모리 할당
   ArrayResize(date,bars);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 선을 그리기 위한 점 지정
   int d=InpDate*(bars-1)/100;
//--- 수직선 생성
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 선을 이동합니다
//--- 루프 카운터
   int h_steps=bars/2;
//--- 선 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d<bars-1)
         d+=1;
      //--- 점 이동
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.03 초 지연
      Sleep(30);
     }
//--- 1초 지연
   Sleep(1000);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 차트에서 채널을 삭제
   VLineDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_HLINE

수평선.

예

다음 스크립트는 차트에서 수평선을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩수평선₩ 그래픽 개체를 그립니다."
#property description "고정점 가격은 다음 높이의 백분율로 설정됩니다"
#property description "차트 창."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="HLine";      // 선 이름
input int             InpPrice=25;          // 선 가격(%)
input color           InpColor=clrRed;      // 선 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 선 스타일
input int             InpWidth=3;           // 선 너비
input bool            InpBack=false;        // 배경 선
input bool            InpSelection=true;    // 이동하려면 강조 표시
input bool            InpHidden=true;       // 개체 목록에 숨겨짐
input long            InpZOrder=0;          // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 수평선 생성                                                      |
//+------------------------------------------------------------------+
bool HLineCreate(const long            chart_ID=0,        // 차트의 ID
```

© 2000-2025, MetaQuotes Ltd.

```
                  const string          name="HLine",      // 선 이름
                  const int             sub_window=0,      // 하위 창 인덱스
                  double                price=0,           // 선 가격
                  const color           clr=clrRed,        // 선 색상
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // 선 스타일
                  const int             width=1,           // 선 너비
                  const bool            back=false,        // 배경에
                  const bool            selection=true,    // 이동하려면 강조 표시
                  const bool            hidden=true,       // 개체 목록에 숨겨짐
                  const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 가격이 설정되지 않은 경우 현재 입찰 가격 수준으로 설정합니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 수평선 생성
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": 수평선 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 선 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 표시 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 선을 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 수평선 이동                                                      |
//+------------------------------------------------------------------+
bool HLineMove(const long   chart_ID=0,   // 차트의 ID
               const string name="HLine", // 선 이름
```

© 2000-2025, MetaQuotes Ltd.

```
                double       price=0)      // 선 가격
  {
//--- 선 가격이 설정되지 않은 경우 현재 입찰 가격 수준으로 이동합니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 수평선 이동
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": 수평선 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 수평선 삭제                                                      |
//+------------------------------------------------------------------+
bool HLineDelete(const long   chart_ID=0,   // 차트의 ID
                 const string name="HLine") // 선 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 수평선 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 수평선 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 가격 값을 저장하기 위한 배열
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 선 고정점 좌표 설정 및 변경
   double price[];
//--- 메모리 할당
   ArrayResize(price,accuracy);
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 선을 그리기 위한 점 지정
   int p=InpPrice*(accuracy-1)/100;
//--- 수평선 생성
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 선을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/2;
//--- 선 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p<accuracy-1)
         p+=1;
      //--- 점 이동
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 삭제
   HLineDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TREND

추세선.

참조

추세선의 경우 우측 및/또는 좌측으로 디스플레이 연속 모드 (OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다.

예

다음 스크립트는 차트에서 추세선을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩추세선₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="Trend";      // 선 이름
input int             InpDate1=35;          // 첫 번째 점 날짜(%)
input int             InpPrice1=60;         // 첫 번째 점 가격(%)
input int             InpDate2=65;          // 두 번째 점 날짜(%)
input int             InpPrice2=40;         // 두 번째 점 가격(%)
input color           InpColor=clrRed;      // 선 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 선 스타일
input int             InpWidth=2;           // 선 너비
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpBack=false;       // 배경 선
input bool            InpSelection=true;   // 이동하려면 강조 표시
input bool            InpRayLeft=false;    // 좌측으로 선 연장
input bool            InpRayRight=false;   // 우측으로 선 연장
input bool            InpHidden=true;      // 개체 목록에 숨겨짐
input long            InpZOrder=0;         // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표 추세선 생성                                          |
//+------------------------------------------------------------------+
bool TrendCreate(const long            chart_ID=0,        // 차트의 ID
                 const string          name="TrendLine",  // 선 이름
                 const int             sub_window=0,      // 하위 창 인덱스
                 datetime              time1=0,           // 첫 번째 점 시간
                 double                price1=0,          // 첫 번째 점 가격
                 datetime              time2=0,           // 두 번째 점 시간
                 double                price2=0,          // 두 번째 점 가격
                 const color           clr=clrRed,        // 선 색상
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // 선 스타일
                 const int             width=1,           // 선 너비
                 const bool            back=false,        // 배경에
                 const bool            selection=true,    // 이동하려면 강조 표시
                 const bool            ray_left=false,    // 좌측으로 선 연장
                 const bool            ray_right=false,   // 우측으로 선 연장
                 const bool            hidden=true,       // 개체 목록에 숨겨짐
                 const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 지정된 좌표로 추세선 작성
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": 추세선 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 선 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 표시 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 선을 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 선 디스플레이의 좌측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 선 디스플레이의 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 추세선 고정점 이동                                               |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // 차트의 ID
                      const string name="TrendLine", // 선 이름
                      const int    point_index=0,    // 고정점 인덱스
                      datetime     time=0,           // 고정점 시간 좌표
                      double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 추세선 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에서 추세선을 삭제하는 기능.                                 |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // 차트의 ID
                 const string name="TrendLine") // 선 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 추세선 삭제
   if(!ObjectDelete(chart_ID,name))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": 추세선 삭제 실패! Error code = ",GetLastError());
       return(false);
      }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 추세선의 고정점 값 확인 기본값 설정   |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- 첫 번째 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time1)
      time1=TimeCurrent();
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 점의 시간이 설정되지 않은 경우 두 번째 점에서 남은 9 막대가 위치합니다
   if(!time2)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 첫 번째 점에서 남은 9개 막대에 두 번째 점을 설정
      time2=temp[0];
     }
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점의 가격과 동일합니다
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
```

© 2000-2025, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 선 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 선을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 추세선 생성
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 선의 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/5;
//--- 첫 번째 고정점을 수직방향으로 이동합니다
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1>1)
         p1-=1;
      //--- 점 이동
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 두 번째 고정점을 수직방향으로 이동합니다
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p2<accuracy-1)
         p2+=1;
      //--- 점 이동
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 0.5초 지연
   Sleep(500);
//--- 루프 카운터
   int h_steps=bars/2;
//--- 두 고정점을 동시에 수평방향으로 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값을 사용
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- 점 변경
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.03 초 지연
      Sleep(30);
     }
//--- 1초 지연
   Sleep(1000);
//--- 추세선 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TRENDBYANGLE

각도별 추세선.

참조

각도별 추세선의 경우 디스플레이의 우측 및/또는 좌측 연속 모드 (OBJPROP_RAY_RIGHT and OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다.

각도 및 두 번째 고정점 좌표를 모두 사용하여 선 기울기를 설정할 수 있습니다.

예

다음 스크립트는 차트에서 추세선을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩각도별 추세선₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 다음 크기의 백분율로 설정됩니다"
#property description "차트 창."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="Trend";      // 선 이름
input int             InpDate1=50;          // 첫 번째 점 날짜(%)
input int             InpPrice1=75;         // 첫 번째 점 가격(%)
input int             InpAngle=0;           // 선 기울기 각도
input color           InpColor=clrRed;      // 선 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 선 스타일
input int             InpWidth=2;           // 선 너비
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpBack=false;       // 배경 선
input bool            InpSelection=true;   // 이동하려면 강조 표시
input bool            InpRayLeft=false;    // 좌측으로 선 연장
input bool            InpRayRight=true;    // 우측으로 선 연장
input bool            InpHidden=true;      // 개체 목록에 숨겨짐
input long            InpZOrder=0;         // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 각도별 추세선 생성                                               |
//+------------------------------------------------------------------+
bool TrendByAngleCreate(const long            chart_ID=0,        // 챁트의 ID
                        const string          name="TrendLine",  // line name
                        const int             sub_window=0,      // 하위 창 인덱스
                        datetime              time=0,            // 점 시간
                        double                price=0,           // 점 가격
                        const double          angle=45.0,        // 기울기 각도
                        const color           clr=clrRed,        // 선 색상
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // 선 스타일
                        const int             width=1,           // 선 너비
                        const bool            back=false,        // 배경에
                        const bool            selection=true,    // 이동하려면 강조 표시
                        const bool            ray_left=false,    // 좌측으로 선 연장
                        const bool            ray_right=true,    // 우측으로 선 연장
                        const bool            hidden=true,       // 개체 목록에 숨겨짐
                        const long            z_order=0)         // 마우스 클릭 우선 순
  {
//--- 마우스로 추세선을 쉽게 끌 수 있는 두 번째 점 생성
   datetime time2=0;
   double   price2=0;
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 두 점을 사용하여 추세선 생성
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2))
     {
      Print(__FUNCTION__,
            ": 추세선 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 두 번째 좌표, 각도 변경 시 추세선의 기울기 각도 변경
//--- 선의 점은 각도의 새 값에 따라 자동으로 재정의됩니다
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- 선 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 선을 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 선 디스플레이의 좌측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 선 디스플레이의 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 추세선 고정점 좌표 변경                                          |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // 차트의 ID
                      const string name="TrendLine", // 선 이름
                      datetime     time=0,           // 고정점 시간 좌표
                      double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 경우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 추세선 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 추세선 기울기 각도 변경                                          |
//+------------------------------------------------------------------+
bool TrendAngleChange(const long   chart_ID=0,       // 차트의 ID
                      const string name="TrendLine", // 추세선 이름
                      const double angle=45)         // 추세선 기울기 각도
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 추세 선 경사각 변경
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": 선 기울기 각도 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 추세선 삭제                                                      |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // 차트의 ID
                 const string name="TrendLine") // 선 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 추세선 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 추세선 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 추세선의 고정점 값 확인 기본값 설정   |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- 첫 번째 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time1)
      time1=TimeCurrent();
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 보조점 좌표 설정
//--- 두 번째 점은 9 막대를 남기고 가격은 같습니다
   datetime second_point_time[10];
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
```

© 2000-2025, MetaQuotes Ltd.

```
   price2=price1;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 선 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 선을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- 추세선 생성
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
   Sleep(1000);
//--- 이제 선을 이동하고 회전합니다
//--- 루프 카운터
   int v_steps=accuracy/2;
//--- 고정점을 이동하고 선의 경사각을 변경
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1>1)
         p1-=1;
      //--- 점 이동
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 삭제
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CYCLES

주기선.

참조

선 사이의 거리는 개체의 두 고정점의 시간 좌표로 설정됩니다.

예

다음 스크립트는 차트에서 주기선을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 차트에 주기선을 생성합니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string           InpName="Cycles";   // 개체 이름
input int              InpDate1=10;        // 첫 번째 점 날짜(%)
input int              InpPrice1=45;       // 첫 번째 점 가격(%)
input int              InpDate2=20;        // 두 번째 점 날짜(%)
input int              InpPrice2=55;       // 두 번째 점 가격(%)
input color            InpColor=clrRed;    // 주기선의 색상
input ENUM_LINE_STYLE  InpStyle=STYLE_DOT; // 주기선의 스타일
input int              InpWidth=1;         // 주기선의 너비
input bool             InpBack=false;      // 배경 개체
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpSelection=true;  // 이동하려면 강조 표시
input bool            InpHidden=true;     // 개체 목록에 숨겨짐
input long            InpZOrder=0;        // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주기선 생성                                                      |
//+------------------------------------------------------------------+
bool CyclesCreate(const long            chart_ID=0,        // 차트의 ID
                  const string          name="Cycles",     // 개체 이름
                  const int             sub_window=0,      // 하위 창 인덱스
                  datetime              time1=0,           // 첫 번째 점 시간
                  double                price1=0,          // 첫 번째 점 가격
                  datetime              time2=0,           // 두 번째 점 시간
                  double                price2=0,          // 두 번째 점 가격
                  const color           clr=clrRed,        // 주기선의 색상
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // 주기선 스타일
                  const int             width=1,           // 주기선의 너비
                  const bool            back=false,        // 배경에
                  const bool            selection=true,    // 이동하려면 강조 표시
                  const bool            hidden=true,       // 개체 목록에 숨겨짐
                  const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표로 주기선 생성
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": f주기선 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 선 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선의 스타일 표시 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선의 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 선을 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool CyclesPointChange(const long   chart_ID=0,    // 차트의 ID
                       const string name="Cycles", // 개체 이름
                       const int    point_index=0, // 고정점 인덱스
                       datetime     time=0,        // 고정점 시간 좌표
                       double       price=0)       // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 주기선 삭제                                                      |
//+------------------------------------------------------------------+
bool CyclesDelete(const long   chart_ID=0,    // 차트의 ID
                  const string name="Cycles") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 주기선 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 주기선 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 주기선의 고정점 값을 확인하고 기본값 설정   |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- 첫 번째 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time1)
      time1=TimeCurrent();
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 점의 시간이 설정되지 않은 경우 두 번째 점에서 남은 9 막대가 위치합니다
   if(!time2)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 첫 번째 점에서 남은 9개 막대에 두 번째 점을 설정
      time2=temp[0];
     }
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점의 가격과 동일합니다
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 주기선의 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 주기선을 그릴 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 추세선 생성
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동합니다
//--- 루프 카운터
   int h_steps=bars/5;
//--- 두 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d2<bars-1)
         d2+=1;
      //--- 점 이동
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
//--- 루프 카운터
   h_steps=bars/4;
//--- 첫 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d1<bars-1)
         d1+=1;
      //--- 점 이동
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 개체 삭제
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROWED_LINE

화살표 선.

예

다음 스크립트는 차트에서 화살표 선을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩화살표 선₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="ArrowedLine"; // 선 이름
input int             InpDate1=35;           // 첫 번째 점 날짜(%)
input int             InpPrice1=60;          // 첫 번째 점 가격(%)
input int             InpDate2=65;           // 두 번째 점 날짜(%)
input int             InpPrice2=40;          // 두 번째 점 가격(%)
input color           InpColor=clrRed;       // 선 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;   // 선 스타일
input int             InpWidth=2;            // 선 너비
input bool            InpBack=false;         // 배경 선
input bool            InpSelection=true;     // 이동하려면 강조 표시
input bool            InpHidden=true;        // 개체 목록에 숨겨짐
input long            InpZOrder=0;           // 마우스 클릭 우선 순위
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 화살표 선 작성                            |
//+------------------------------------------------------------------+
bool ArrowedLineCreate(const long            chart_ID=0,         // 차트의 ID
                       const string          name="ArrowedLine", // 선 이름
                       const int             sub_window=0,       // 하위 창 인덱스
                       datetime              time1=0,            // 첫 번째 점 시간
                       double                price1=0,           // 첫 번째 점 가격
                       datetime              time2=0,            // 두 번째 점 시간
                       double                price2=0,           // 두 번째 점 가격
                       const color           clr=clrRed,         // 선 색상
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // 선 스타일
                       const int             width=1,            // 선 굵기
                       const bool            back=false,         // 배경에
                       const bool            selection=true,     // 이동하려면 강조 표시
                       const bool            hidden=true,        // 개체 목록에 숨겨짐
                       const long            z_order=0)          // 마우스 클릭 우선 순
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표로 화살표 선을 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price
     {
      Print(__FUNCTION__,
            ": 화살표 선 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 선 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 표시 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 선을 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 화살표 선의 고정점 이동                                          |
//+------------------------------------------------------------------+
bool ArrowedLinePointChange(const long   chart_ID=0,          // 차트의 ID
                            const string name="ArrowedLine", // line name
                            const int    point_index=0,      // 고정점 인덱스
                            datetime     time=0,             // 고정점 시간 좌표
                            double       price=0)            // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 선의 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에서 함수가 화살표 선을 제거합니다                           |
//+------------------------------------------------------------------+
bool ArrowedLineDelete(const long   chart_ID=0,         // 차트의 ID
                       const string name="ArrowedLine") // 선 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 화살표 선 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 화살표 선 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                                    |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- 첫 번째 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time1)
      time1=TimeCurrent();
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 점의 시간이 설정되지 않은 경우 두 번째 점에서 남은 9 막대가 위치합니다
   if(!time2)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 첫 번째 점에서 남은 9개 막대에 두 번째 점을 설정
      time2=temp[0];
     }
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점의 가격과 동일합니다
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 선 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 선을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 화살표 선 생성
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 선의 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/5;
//--- 두 번째 고정점을 수직방향으로 이동합니다
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p2<accuracy-1)
         p2+=1;
      //--- 점 이동
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 첫 번째 고정점을 수직방향으로 이동합니다
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1>1)
         p1-=1;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 점 이동
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 0.5초 지연
   Sleep(500);
//--- 루프 카운터
   int h_steps=bars/2;
//--- 두 고정점을 동시에 수평방향으로 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값을 사용
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- 점 변경
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.03 초 지연
      Sleep(30);
     }
//--- 1초 지연
   Sleep(1000);
//--- 화살표 선 삭제
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CHANNEL

등거리 채널

참조

등거리 채널의 경우 우측 및/또는 좌측으로 연속 모드 (OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다. 채널 색상 채우기 모드도 설정할 수 있습니다.

예

다음 스크립트는 차트에서 등거리 채널을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩등거리 채널₩ 그래픽 개체를그립니다."
#property description "고정점 좌표는 다음 크기의 백분율로 설정됩니다"
#property description "차트 창."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="Channel";   // 채널 이름
input int             InpDate1=25;         // 첫 번째 점 날짜(%)
input int             InpPrice1=60;        // 첫 번째 점 가격(%)
input int             InpDate2=65;         // 두 번째 점 날짜(%)
input int             InpPrice2=80;        // 두 번째 점 가격(%)
input int             InpDate3=30;         // 세 번째 점 날짜(%)
input int             InpPrice3=40;        // 세 번째 점 가격(%)
```

© 2000-2025, MetaQuotes Ltd.

```
input color           InpColor=clrRed;      // 채널 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 채널 선의 스타일
input int             InpWidth=2;           // 채널 선 너비
input bool            InpBack=false;        // 배경 채널
input bool            InpFill=false;        // 채널을 컬러로 채움
input bool            InpSelection=true;    // 이동하려면 강조 표시
input bool            InpRayLeft=false;     // 채널 좌측으로 연장
input bool            InpRayRight=false;    // 채널 우측으로 연장
input bool            InpHidden=true;       // 개체 목록에 숨겨짐
input long            InpZOrder=0;          // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 등거리 채널 생성                          |
//+------------------------------------------------------------------+
bool ChannelCreate(const long             chart_ID=0,        // 차트의 ID
                   const string           name="Channel",    // 채널 이름
                   const int              sub_window=0,      // 하위 창 인덱스
                   datetime               time1=0,           // 첫 번째 점 시간
                   double                 price1=0,          // 첫 번째 점 가격
                   datetime               time2=0,           // 두 번째 점 시간
                   double                 price2=0,          // 두 번째 점 가격
                   datetime               time3=0,           // 세 번째 점 시간
                   double                 price3=0,          // 세 번째 점 가격
                   const color            clr=clrRed,        // 채널 색상
                   const ENUM_LINE_STYLE  style=STYLE_SOLID, // 채널 선 스타일
                   const int              width=1,           // 채널 선 너비
                   const bool             fill=false,        // 채널 색상 채우기
                   const bool             back=false,        // 배경에
                   const bool             selection=true,    // 이동하려면 강조 표시
                   const bool             ray_left=false,    // 채널 좌측으로 연장
                   const bool             ray_right=false,   // 채널 우측으로 연장
                   const bool             hidden=true,       // 개체 목록에 숨겨짐
                   const long             z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표로 채널 생성
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": 등거리 채널 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 채널 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 채널 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 채널 선 너비 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 채널 채우기 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 채널 디스플레이의 좌측 연속 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 채널 디스플레이 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 채널의 고정점 이동                                               |
//+------------------------------------------------------------------+
bool ChannelPointChange(const long   chart_ID=0,     // 차트의 ID
                        const string name="Channel", // 채널 이름
                        const int    point_index=0,  // 고정점 인덱스
                        datetime     time=0,         // 고정점 시간 좌표
                        double       price=0)        // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 경우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 채널 삭제                                                        |
//+------------------------------------------------------------------+
bool ChannelDelete(const long   chart_ID=0,     // 차트의 ID
                   const string name="Channel") // 채널 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 채널 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 채널 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------------+
//| 채널의 고정점 값을 확인하고 기본값을 설정  |
//| 빈 곳의 경우                                                           |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- 두 번째(우측) 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 두 번째 점의 가격이 설정되지 않은 경우 입찰 값이 지정됩니다
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 첫 번째(남은) 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9개 막대에 위치합니다.
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 포인트의 가격이 설정되지 않은 경우 두 번째 포인트보다 300 포인트 높게 이동합니다
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 세 번째 점의 시간이 설정되지 않은 경우 첫 번째 점의 시간과 일치합니다
   if(!time3)
      time3=time1;
//--- 세 번째 점의 가격이 설정되지 않은 경우 두 번째 점의 가격과 같습니다
   if(!price3)
      price3=price2;
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 채널 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 채널을 그리기 위한 점 정의
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 등거리 채널 생성
   if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       return;
      }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 채널의 고정점을 이동합니다
//--- 루프 카운터
   int h_steps=bars/6;
//--- 두 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d2<bars-1)
         d2+=1;
      //--- 점 이동
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 첫 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d1>1)
         d1-=1;
      //--- 점 이동
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int v_steps=accuracy/10;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 세 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p3>1)
         p3-=1;
      //--- 점 이동
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 채널을 삭제
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_STDDEVCHANNEL

표준 편차 채널.

참조

표준 편차 채널의 경우 우측 및/또는 좌측으로 디스플레이 연속 모드 (OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다. 채널 색상 채우기 모드도 설정할 수 있습니다.

OBJPROP_DEVIATION 속성은 채널 편차 값을 변경하는 데 사용됩니다.

예

다음 스크립트는 차트에서 표준 편차 채널을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩표준 편차 채널₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 다음 크기의 백분율로 설정됩니다"
#property description "차트 창."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="StdDevChannel";    // 채널 이름
input int             InpDate1=10;                // 첫 번째 점 날짜(%)
input int             InpDate2=40;                // 두 번째 점 날짜(%)
input double          InpDeviation=1.0;           // 편차
input color           InpColor=clrRed;            // 채널 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 채널 선 스타일
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpWidth=2;                // 채널 선 너비
input bool             InpFill=false;             // 채널 색상 채우기
input bool             InpBack=false;             // 배경 채널
input bool             InpSelection=true;         // 이동하려면 강조 표시
input bool             InpRayLeft=false;          // 채널 좌측으로 연장
input bool             InpRayRight=false;         // 채널 우측으로 연장
input bool             InpHidden=true;            // 개체 목록에 숨겨짐
input long             InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 표준 편차 채널 생성                       |
//+------------------------------------------------------------------+
bool StdDevChannelCreate(const long            chart_ID=0,        // 차트의 ID
                         const string          name="Channel",    // 채널 이름
                         const int             sub_window=0,      // 하위 창 인덱스
                         datetime              time1=0,           // 첫 번째 점 시간
                         datetime              time2=0,           // 두 번째 점 시간
                         const double          deviation=1.0,     // 편차
                         const color           clr=clrRed,        // 채널 색상
                         const ENUM_LINE_STYLE style=STYLE_SOLID, // 채널 선 스타일
                         const int             width=1,           // 채널 선 너비
                         const bool            fill=false,        // 채널 색상 채우기
                         const bool            back=false,        // 배경에
                         const bool            selection=true,    // 이동하려면 강조 표시
                         const bool            ray_left=false,    // 좌측으로 채널 연속
                         const bool            ray_right=false,   // 우측으로 채널 연속
                         const bool            hidden=true,       // 개체 목록에 숨겨짐
                         const long            z_order=0)         // 마우스 클릭 우선 순
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeChannelEmptyPoints(time1,time2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표로 채널 생성
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": 표준 편차 채널 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 채널 너비에 영향을 미치는 편차 값 설정
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- 채널 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 채널 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 채널 선 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 채널 채우기 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 채널 디스플레이의 좌측 연속 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 채널 디스플레이 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 채널의 고정점 이동                                               |
//+------------------------------------------------------------------+
bool StdDevChannelPointChange(const long   chart_ID=0,     // 차트의 ID
                              const string name="Channel", // 채널 이름
                              const int    point_index=0,  // 고정점 인덱스
                              datetime     time=0)         // 고정점 시간 좌표
  {
//--- 점 시간이 설정되지 않은 경우 점을 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 채널 편차 변경                                                   |
//+------------------------------------------------------------------+
bool StdDevChannelDeviationChange(const long   chart_ID=0,     // 차트의 ID
                                  const string name="Channel", // 채널 이름
                                  const double deviation=1.0)  // 편차
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 값 재설정
   ResetLastError();
//--- 추세 선 경사각 변경
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            ": 채널 편차 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 채널 삭제                                                        |
//+------------------------------------------------------------------+
bool StdDevChannelDelete(const long   chart_ID=0,     // 차트의 ID
                         const string name="Channel") // 채널 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 채널 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 채널 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------------+
//| 채널의 고정점 값을 확인하고 기본값을 설정  |
//| 빈 곳의 경우                                                           |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- 두 번재 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 첫 번째 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9 막대가 위치합니다
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 채널 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 채널을 그리기 위한 점 정의
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- 표준 편차 채널 생성
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpStyl
      InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
//--- 이제 채널을 우측 수평방향으로 이동하여 확장합니다
//--- 루프 카운터
   int h_steps=bars/2;
//--- 채널 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값을 사용
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- 고정점 이동
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   double v_steps=InpDeviation*2;
//--- 채널 확장
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 채널을 삭제
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_REGRESSION

선형 회귀 채널.

참조

선형 회귀 채널의 경우 우측 및/또는 좌측으로 디스플레이 연속 모드 (OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다. 채널 색상 채우기 모드도 설정할 수 있습니다.

예

다음 스크립트는 차트에서 선형 회귀 채널을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩선형 회귀 채널₩ 그래픽 개체 를 그립니다."
#property description "고정점 좌표는 다음 크기의 백분율로 설정됩니다"
#property description "차트 창."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="Regression"; // 채널 이름
input int             InpDate1=10;          // 첫 번째 점 날짜(%)
input int             InpDate2=40;          // 두 번째 점 날짜(%)
input color           InpColor=clrRed;      // 채널 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // 채널 선 스타일
input int             InpWidth=2;           // 채널 선 너비
input bool            InpFill=false;        // 채널 색상 채우기
```

© 2000-2025, MetaQuotes Ltd.

```
input bool             InpBack=false;         // 배경 채널
input bool             InpSelection=true;     // 이동하려면 강조 표시
input bool             InpRayLeft=false;      // 좌측으로 채널 연장
input bool             InpRayRight=false;     // 우측으로 채널 연장
input bool             InpHidden=true;        // 개체 목록에 숨겨짐
input long             InpZOrder=0;           // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 선형 회귀 채널 생성                       |
//+------------------------------------------------------------------+
bool RegressionCreate(const long             chart_ID=0,        // 차트의 ID
                      const string           name="Regression", // 채널 이름
                      const int              sub_window=0,      // 하위 창 인덱스
                      datetime               time1=0,           // 첫 번째 점 시간
                      datetime               time2=0,           // 두 번째 점 시간
                      const color            clr=clrRed,        // 채널 색상
                      const ENUM_LINE_STYLE  style=STYLE_SOLID, // 채널 선 스타일
                      const int              width=1,           // 채널 선 너비
                      const bool             fill=false,        // 채널 색상 채우기
                      const bool             back=false,        // 배경에
                      const bool             selection=true,    // 이동하려면 강조 표시
                      const bool             ray_left=false,    // 좌측으로 채널 연속
                      const bool             ray_right=false,   // 우측으로 채널 연속
                      const bool             hidden=true,       // 개체 목록에 숨겨짐
                      const long             z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeRegressionEmptyPoints(time1,time2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표로 채널 생성
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": 선형 회귀 채널 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 채널 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 채널 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 채널 선 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 채널 채우기 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 채널 디스플레이의 좌측 연속 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 채널 디스플레이 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 채널의 고정점 이동                                               |
//+------------------------------------------------------------------+
bool RegressionPointChange(const long   chart_ID=0,     // 차트의 ID
                           const string name="Channel", // 채널 이름
                           const int    point_index=0,  // 고정점 인덱스
                           datetime     time=0)         // 고정점 시간 좌표
  {
//--- 점 시간이 설정되지 않은 경우 점을 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 채널 삭제                                                        |
//+------------------------------------------------------------------+
bool RegressionDelete(const long   chart_ID=0,     // 차트의 ID
                      const string name="Channel") // 채널 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 채널 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": 채널 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------------+
//| 채널의 고정점 값을 확인하고 기본값을 설정  |
//| 빈 곳의 경우                                                           |
//+------------------------------------------------------------------------+
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- 두 번재 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 첫 번째 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9 막대가 위치합니다
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 채널 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 채널을 그리기 위한 점 정의
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- 선형 회귀 채널 생성
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 채널을 수평방향으로 우측 이동합니다
//--- 루프 카운터
   int h_steps=bars/2;
//--- 채널 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값을 사용
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- 고정점 이동
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 채널을 삭제
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_PITCHFORK

앤드류스 피치포크.

참조

앤드류스 피치포크의 경우 우측 및/또는 좌측으로 디스플레이 연속 모드(OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다.

선 레벨 숫자, 선 수준 값 및 색상을 지정할 수도 있습니다.

예

다음 스크립트는 차트에서 앤드류스 피치포크를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩앤드류스 피치포크₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="Pitchfork";  // 피치포크 이름
input int             InpDate1=14;          // 첫 번째 점 날짜(%)
input int             InpPrice1=40;         // 첫 번째 점 가격(%)
input int             InpDate2=18;          // 두 번째 점 날짜(%)
input int             InpPrice2=50;         // 두 번째 점 가격(%)
input int             InpDate3=18;          // 세 번째 점 날짜(%)
input int             InpPrice3=30;         // 세 번째 점 가격(%)
```

© 2000-2025, MetaQuotes Ltd.

```
input color           InpColor=clrRed;      // 피치포크 색상
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // 피치포크 선 스타일
input int             InpWidth=1;           // 피치포크 선 너비
input bool            InpBack=false;        // 배경 피치포크
input bool            InpSelection=true;    // 이동하려면 강조 표시
input bool            InpRayLeft=false;     // 좌측으로 피치포크 연장Pitchfork's continua
input bool            InpRayRight=false;    // 우측으로 피치포크 연장
input bool            InpHidden=true;       // 개체 목록에 숨겨짐
input long            InpZOrder=0;          // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 앤드류스 피치포크를 생성                  |
//+------------------------------------------------------------------+
bool PitchforkCreate(const long            chart_ID=0,        // 차트의 ID
                     const string          name="Pitchfork",  // 피치포크 이름
                     const int             sub_window=0,      // 하위 창 인덱스
                     datetime              time1=0,           // 첫 번째 점 시간
                     double                price1=0,          // 첫 번째 점 가격
                     datetime              time2=0,           // 두 번째 점 시간
                     double                price2=0,          // 두 번째 점 가격
                     datetime              time3=0,           // 세 번째 점 시간
                     double                price3=0,          // 세 번째 점 가격
                     const color           clr=clrRed,        // 피치포크 선 색상
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // 피치포크 선 스타일
                     const int             width=1,           // 피치포크 선 너비
                     const bool            back=false,        // 배경에
                     const bool            selection=true,    // 이동하려면 강조 표시
                     const bool            ray_left=false,    // 좌측으로 피치포크 연장
                     const bool            ray_right=false,   // 우측으로 피치포크 연장
                     const bool            hidden=true,       // 개체 목록에 숨겨짐
                     const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 앤드류스 피치포크를 생성
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": \"앤드류스 피치포크₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선의 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동을 위해 피치포크를 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 피치포크 디스플레이 좌측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 피치포크 디스플레이 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 앤드류스 피치포크 레벨 및 해당 매개변수 설정     |
//+------------------------------------------------------------------+
bool PitchforkLevelsSet(int              levels,           // 레벨 선 수
                        double           &values[],        // 레벨 선 값
                        color            &colors[],        // 레벨 선 색상
                        ENUM_LINE_STYLE  &styles[],        // 레벨 선 스타일
                        int              &widths[],        // 레벨 선 너비
                        const long       chart_ID=0,       // 차트의 ID
                        const string     name="Pitchfork") // 피치포크 이름
  {
//--- 배열 크기 확인
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": 오류! 배열 길이가 레벨 숫자와 일치하지 않습니다.");
      return(false);
     }
//--- 레벨 숫자 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- 루프에서 레벨 속성 설정
   for(int i=0;i<levels;i++)
     {
      //--- 레벨 값
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- 레벨 색상
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- 레벨 스타일
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- 레벨 너비
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 레벨 설명
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 앤드류스 피치포크 고정점 이동                                    |
//+------------------------------------------------------------------+
bool PitchforkPointChange(const long   chart_ID=0,       // 차트의 ID
                          const string name="Pitchfork", // 채널 이름
                          const int    point_index=0,    // 고정점 인덱스
                          datetime     time=0,           // 고정점 시간 좌표
                          double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 앤드류스 피치포크 삭제                                           |
//+------------------------------------------------------------------+
bool PitchforkDelete(const long   chart_ID=0,       // 차트의 ID
                     const string name="Pitchfork") // 채널 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 채널 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"앤드류스 피치포크₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 앤드류스 피치포크 고정점 값을 확인하고 기본값 설정 |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- 두 번째(우측 상단) 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 두 번째 점의 가격이 설정되지 않은 경우 입찰 값이 지정됩니다
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 첫 번째(남은) 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9개 막대에 위치합니다.
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 점의 가격이 설정되지 않은 경우 두 번째 점보다 200 포인트 낮게 이동합니다
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 세 번째 점의 시간이 설정되지 않은 경우 두 번째 점의 시간과 일치합니다
   if(!time3)
      time3=time2;
//--- 세 번째 점의 가격이 정해지지 않은 경우 첫 번째 점보다 200 포인트 낮게 이동합니다
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
```

© 2000-2025, MetaQuotes Ltd.

```
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 앤드류스 피치포크 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 앤드류스 피치포크를 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 피치포크 생성
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 피치포크 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/10;
//--- 첫 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1>1)
         p1-=1;
      //--- 점 이동
```

© 2000-2025, MetaQuotes Ltd.

```
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int h_steps=bars/8;
//--- 세 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d3<bars-1)
         d3+=1;
      //--- 점 이동
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy/10;
//--- 두 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p2>1)
         p2-=1;
      //--- 점 이동
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 피치포크 삭제
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNLINE

갠 선(Gann Line).

참조

갠 선의 경우 우측 및/또는 좌측으로 디스플레이의 연속 모드((OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다.

두 번째 고정점의 스케일과 좌표가 있는 갠 각도를 사용하여 선의 기울기를 설정할 수 있습니다.

예

다음 스크립트는 차트에서 갠 선을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩갠 선₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="GannLine";        // 선 이름
input int             InpDate1=20;               // 첫 번째 점 날짜(%)
input int             InpPrice1=75;              // 첫 번째 점 가격(%)
input int             InpDate2=80;               // 두 번째 점 날짜(%)
input double          InpAngle=0.0;              // 갠 각도
input double          InpScale=1.0;              // 스케일
input color           InpColor=clrRed;           // 선 색상
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 선 스타일
input int             InpWidth=2;                // 선 너비
input bool            InpBack=false;             // 배경 선
input bool            InpSelection=true;         // 이동하려면 강조 표시
input bool            InpRayLeft=false;          // 좌측으로 선 연장
input bool            InpRayRight=true;          // 우측으로 선 연장
input bool            InpHidden=true;            // 개체 목록에 숨겨짐
input long            InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표, 각도 및 스케일로 갠 그리드 생성                     |
//+------------------------------------------------------------------+
bool GannLineCreate(const long            chart_ID=0,        // 차트의 ID
                    const string          name="GannLine",   // 선 이름
                    const int             sub_window=0,      // 하위 창 인덱스
                    datetime              time1=0,           // 첫 번째 점 시간
                    double                price1=0,          // 첫 번째 점 가격
                    datetime              time2=0,           // 두 번째 점 시간
                    const double          angle=1.0,         // 갠 각도
                    const double          scale=1.0,         // 스케일
                    const color           clr=clrRed,        // 선 색상
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // 선 스타일
                    const int             width=1,           // 선 너비
                    const bool            back=false,        // 배경에
                    const bool            selection=true,    // 이동하려면 강조 표시
                    const bool            ray_left=false,    // 좌측으로 선 연장
                    const bool            ray_right=true,    // 우측으로 선 연장
                    const bool            hidden=true,       // 개체 목록에 숨겨짐
                    const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 갠 선 생성
//--- 두 번째 고정점의 정확한 좌표가 재정의 됨
//--- 자동으로 갠 각도 및/또는 척도가 변경되면,
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": \"갠 선₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 갠 각도 변경
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- 스케일(막대당 pips 수) 변경
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- 선 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 표시 스타일 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동하기 위해 선을 강조 표시 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 선 디스플레이의 좌측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 선 디스플레이의 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 선 고정점 이동 point                                          |
//+------------------------------------------------------------------+
bool GannLinePointChange(const long   chart_ID=0,       // 차트의 ID
                         const string name="GannLine", // 선 이름
                         const int    point_index=0,   // 고정점 인덱스
                         datetime     time=0,          // 고정점 시간 좌표
                         double       price=0)         // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 선의 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 갠 각도 변경                                                     |
//+------------------------------------------------------------------+
bool GannLineAngleChange(const long   chart_ID=0,      // 차트의 ID
                         const string name="GannLine", // 선 이름
                         const double angle=1.0)       // 갠 각도
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 갠 각도 변경
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": 갠 각도 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 선 스케일 변경                                                |
//+------------------------------------------------------------------+
bool GannLineScaleChange(const long   chart_ID=0,      // 차트의 ID
                         const string name="GannLine", // 선 이름
                         const double scale=1.0)       // 스케일
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 스케일(막대당 pips 수) 변경
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": 스케일 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에서 갠 선 제거 기능                                         |
//+------------------------------------------------------------------+
bool GannLineDelete(const long   chart_ID=0,      // 차트의 ID
                    const string name="GannLine") // 선 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 갠 선 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": \"갠 선₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 선 고정점 값을 확인하고 기본 값을 설정      |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- 두 번재 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 첫 번째 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9 막대가 위치합니다
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 선 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
```

© 2000-2025, MetaQuotes Ltd.

```
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 갠 선을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- 갠 선 생성
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpCol
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 선의 고정점을 이동하고 각도를 변경합니다
//--- 루프 카운터
   int v_steps=accuracy/2;
//--- 첫 번째 고정점을 수직방향으로 이동합니다
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1>1)
         p1-=1;
      //--- 점 이동
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 0.5초 지연
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(500);
//--- 갠 각도의 현재 값 지정(변경됨-
//--- 첫 번째 고정점 이동 후)
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//--- 루프 카운터
   v_steps=accuracy/8;
//--- 갠 각도 변경
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 선 삭제
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNFAN

갠 팬(Gann Fan).

참조

갠 팬의 경우 ENUM_GANN_DIRECTION 열거에서 추세 유형을 지정할 수 있습니다. 스케일 값(OBJPROP_SCALE)을 조정하여 팬 선의 기울기 각도를 변경할 수 있습니다.

예

다음 스크립트는 차트에서 갠 팬을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩갠 팬₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="GannFan";          // 팬 이름
input int             InpDate1=15;                // 첫 번째 점 날짜(%)
input int             InpPrice1=25;               // 첫 번째 점 가격(%)
input int             InpDate2=85;                // 두 번째 점 날짜(%)
input double          InpScale=2.0;               // 스케일
input bool            InpDirection=false;         // 추세 방향
input color           InpColor=clrRed;            // 팬 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 팬 선 스타일
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpWidth=1;                // 팬 선 너비
input bool             InpBack=false;             // 배경 팬
input bool             InpSelection=true;         // 이동하려면 강조 표시
input bool             InpHidden=true;            // 개체 목록에 숨겨짐
input long             InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 갠 팬 생성                                                       |
//+------------------------------------------------------------------+
bool GannFanCreate(const long            chart_ID=0,        // 차트의 ID
                   const string          name="GannFan",    // 팬 이름
                   const int             sub_window=0,      // 하위 창 인덱스
                   datetime              time1=0,           // 첫 번째 점 시간
                   double                price1=0,          // 첫 번째 점 가격
                   datetime              time2=0,           // 두 번째 점 시간
                   const double          scale=1.0,         // 스케일
                   const bool            direction=true,    // 추세 방향
                   const color           clr=clrRed,        // 팬 색상
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // 팬 선 스타일
                   const int             width=1,           // 팬 선 너비
                   const bool            back=false,        // 배경에
                   const bool            selection=true,    // 이동하려면 강조 표시
                   const bool            hidden=true,       // 개체 목록에 숨겨짐
                   const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 갠 팬 생성
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": \"갠 팬₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 스케일(막대당 pips 수) 변경
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- 갠 팬의 추세 방향 변경(true - 내림차순, false - 오름차순)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- 팬 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 팬 선 스타일 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 팬 선 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 팬을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 팬 고정점 이동                                                |
//+------------------------------------------------------------------+
bool GannFanPointChange(const long   chart_ID=0,     // 차트의 ID
                        const string name="GannFan", // 팬 이름
                        const int    point_index=0,  // 고정점 인덱스
                        datetime     time=0,         // 고정점 시간 좌표
                        double       price=0)        // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 경우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 팬의 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 팬 스케일 변경                                                |
//+------------------------------------------------------------------+
bool GannFanScaleChange(const long   chart_ID=0,     // 차트의 ID
                        const string name="GannFan", // 팬 이름
                        const double scale=1.0)      // 스케일
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 스케일(막대당 pips 수) 변경
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": 스케일 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 팬 추세 방향 변경                                             |
//+------------------------------------------------------------------+
bool GannFanDirectionChange(const long   chart_ID=0,     // 차트의 ID
                            const string name="GannFan", // 팬 이름
                            const bool   direction=true) // 추세 방향
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 갠 팬 추세 방향 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": 추세 방향 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이 기능은 차트에서 갠 팬을 제거합니다                            |
//+------------------------------------------------------------------+
bool GannFanDelete(const long   chart_ID=0,     // 차트의s ID
                   const string name="GannFan") // 팬 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 갠 팬 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"갠 팬₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
// | 갠팬 고정점 값 확인 및 기본 값 설정      |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 두 번재 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 첫 번째 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9 막대가 위치합니다
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 팬 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
```

© 2000-2025, MetaQuotes Ltd.

```
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 갠 팬을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- 갠 팬 생성
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 팬의 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/2;
//--- 첫 번째 고정점을 수직방향으로 이동합니다
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1<accuracy-1)
         p1+=1;
      //--- 점 이동
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 팬의 추세 방향을 내림차순으로 변경
   GannFanDirectionChange(0,InpName,true);
//--- 차트 다시 그리기
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//--- delete the fan from the chart
   GannFanDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
```

© 2000-2025, MetaQuotes Ltd.

```
    Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_GANNGRID

갠 그리드(Gann Grid).

참조

갠 그리드의 경우 ENUM_GANN_DIRECTION 추세 유형을 지정할 수 있습니다. 스케일 값(OBJPROP_SCALE)을 조정하여 그리드 선의 기울기 각도를 변경할 수 있습니다.

예

다음 스크립트는 차트에서 갠 그리드를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩갠 그리드₩ 그래픽 개체를 그립니다."
#property description "그리드의 고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="GannGrid";        // 그리드 이름
input int             InpDate1=15;               // 첫 번째 점 날짜(%)
input int             InpPrice1=25;              // 첫 번째 점 가격(%)
input int             InpDate2=35;               // 두 번째 점 날짜(%)
input double          InpScale=3.0;              // 스케일
input bool            InpDirection=false;        // 추세 방향
input color           InpColor=clrRed;           // 그리드 색상
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 그리드 선 스타일
input int             InpWidth=1;                // 팬 선 너비
input bool            InpBack=false;             // 배경 그리드
input bool            InpSelection=true;         // 이동하려면 강조 표시
input bool            InpHidden=true;            // 개체 목록에 숨겨짐
input long            InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 갠 그리드 생성                                                   |
//+------------------------------------------------------------------+
bool GannGridCreate(const long            chart_ID=0,        // 차트의 ID
                    const string          name="GannGrid",   // 그리드 이름
                    const int             sub_window=0,      // 하위 창 인덱스
                    datetime              time1=0,           // 첫 번째 점 시간
                    double                price1=0,          // 첫 번째 점 가격
                    datetime              time2=0,           // 두 번째 점 시간
                    const double          scale=1.0,         // 스케일
                    const bool            direction=true,    // 추세 방향
                    const color           clr=clrRed,        // 그리드 색상
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // 그리드 선 스타일
                    const int             width=1,           // 그리드 선 너비
                    const bool            back=false,        // 배경에
                    const bool            selection=true,    // 이동하려면 강조 표시
                    const bool            hidden=true,       // 개체 목록에 숨겨짐
                    const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 갠 그리드 설정
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": \"갠 그리드₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 스케일(막대당 pips 수) 변경
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- 갠 팬의 추세 방향 변경(true - 내림차순, false - 오름차순)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- 그리드 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 그리드 선 스타일 표시 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 그리드 선 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동을 위해 그리드를 강조 표시하는 모드 활성화(true) 또는 비활성화(false)
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 그리드 고정점 이동                                            |
//+------------------------------------------------------------------+
bool GannGridPointChange(const long   chart_ID=0,      // 차트의 ID
                         const string name="GannGrid", // 그리드 이름
                         const int    point_index=0,   // 고정점 인덱스
                         datetime     time=0,          // 고정점 시간 좌표
                         double       price=0)         // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 그리드의 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 그리드 스케일 변경                                            |
//+------------------------------------------------------------------+
bool GannGridScaleChange(const long   chart_ID=0,      // 차트의 ID
                         const string name="GannGrid", // 그리드
                         const double scale=1.0)       // 스케일
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 스케일(막대당 pips 수) 변경
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": 스케일 변경 실패! Error code = ",GetLastError());
       return(false);
      }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 그리드 추세 방향 변경                                         |
//+------------------------------------------------------------------+
bool GannGridDirectionChange(const long   chart_ID=0,      // 차트의 ID
                             const string name="GannGrid", // 그리드 이름
                             const bool   direction=true)  // 추세 방향
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 갠 그리드 추세 방향 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": 추세 방향 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이 기능은 차트에서 갠 팬을 제거합니다                            |
//+------------------------------------------------------------------+
bool GannGridDelete(const long   chart_ID=0,      // 차트의 ID
                    const string name="GannGrid") // 그리드 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 갠 그리드 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"갠 그리드₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 갠 그리드 고정점 값을 확인하고 기본 값을 설정      |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- 두 번재 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 첫 번째 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9 막대가 위치합니다
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 그리드 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 갠 그리드를 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- 갠 그리드 생성
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 그리드의 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/4;
//--- 첫 번째 고정점을 수직방향으로 이동합니다
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int h_steps=bars/4;
//--- 두 번째 고정점을 수평방향으로 이동합니다
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
         return;
```

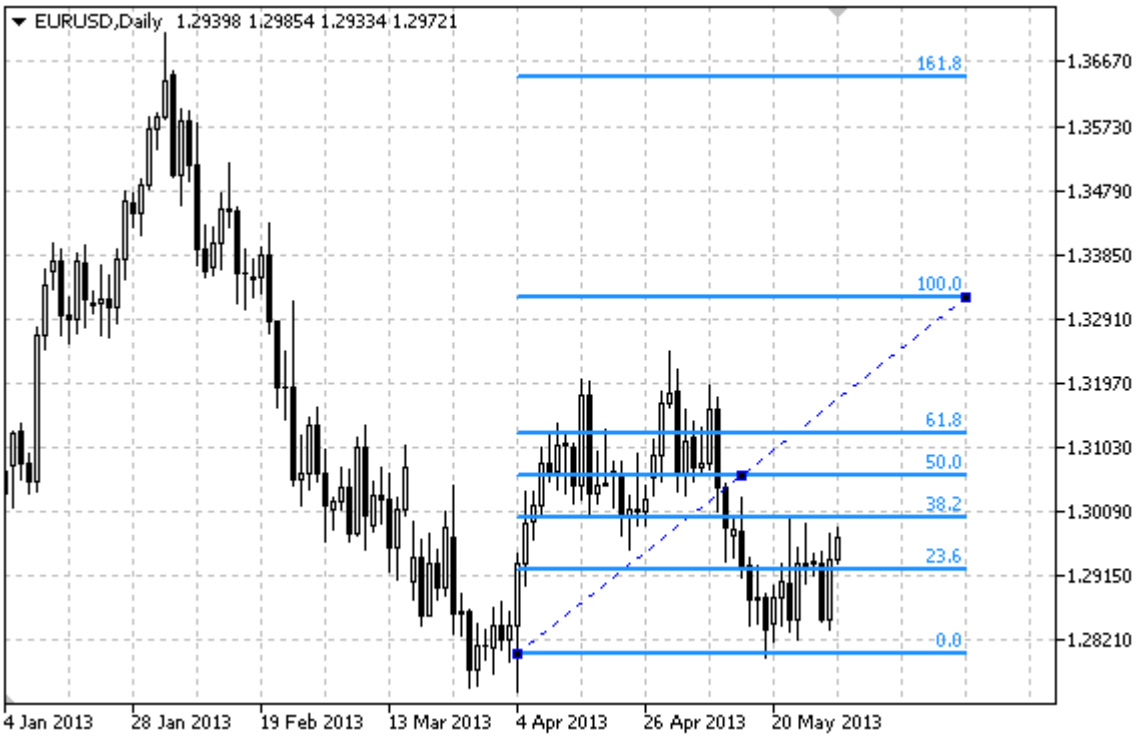
© 2000-2025, MetaQuotes Ltd.

```
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 그리드 추세 방향을 내림차순으로 변경
   GannGridDirectionChange(0,InpName,true);
//--- 차트 다시 그리기
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 그리드를 삭제
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBO

피보나치 되돌림.



참조

피보나치 되돌림의 경우 디스플레이의 우측 및/또는 좌측 연속 모드 (OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다.

선 레벨 숫자, 선 수준 값 및 색상을 지정할 수도 있습니다.

예

다음 스크립트는 차트에서 피보나치 되돌림을 만들고 이동합니다 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩피보나치 되돌림₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="FiboLevels";      // 개체 이름
input int             InpDate1=10;               // 첫 번째 점 날짜(%)
input int             InpPrice1=65;              // 첫 번째 점 가격(%)
input int             InpDate2=90;               // 두 번째 점 날짜(%)
input int             InpPrice2=85;              // 두 번째 점 가격(%)
input color           InpColor=clrRed;           // 개체 색상
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 선 스타일
input int             InpWidth=2;                // 선 너비
input bool            InpBack=false;             // 배경에
input bool            InpSelection=true;         // 이동하려면 강조 표시
input bool            InpRayLeft=false;          // 개체 좌측으로 연장
input bool            InpRayRight=false;         // 개체 우측으로 연장
input bool            InpHidden=true;            // 개체 목록에 숨겨짐
input long            InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 피보나치 되돌림 생성                      |
//+------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // 차트의 ID
                      const string          name="FiboLevels", // 개체 이름
                      const int             sub_window=0,      // 하위 창 인덱스
                      datetime              time1=0,           // 첫 번째 점 시간
                      double                price1=0,          // 첫 번째 점 가격
                      datetime              time2=0,           // 두 번째 점 시간
                      double                price2=0,          // 두 번째 점 가격
                      const color           clr=clrRed,        // 개체 색상
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // 개체 선 스타일
                      const int             width=1,           // 개체 선 너비
                      const bool            back=false,        // 배경에
                      const bool            selection=true,    // 이동하려면 강조 표시
                      const bool            ray_left=false,    // 개체 좌측으로 계속
                      const bool            ray_right=false,   // 개체 우측으로 계속
                      const bool            hidden=true,       // 개체 목록에 숨겨짐
                      const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 피보나치 되돌림 생성
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": \"피보나치 되돌림₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 디스플레이 좌측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 개체 디스플레이의 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 레벨 숫자 및 레벨 매개변수 설정                                  |
//+------------------------------------------------------------------+
bool FiboLevelsSet(int              levels,           // 레벨 선 수
                   double           &values[],        // 레벨 선 값
                   color            &colors[],        // 레벨 선 색상
                   ENUM_LINE_STYLE &styles[],         // 레벨 선 스타일
                   int              &widths[],        // 레벨 선 너비
                   const long       chart_ID=0,       // 차트의 ID
                   const string     name="FiboLevels") // 개체 이름
  {
//--- 배열 크기 확인
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) || levels!=ArraySize(widt
     {
      Print(__FUNCTION__,": 오류! 배열 길이가 레벨 숫자와 일치하지 않습니다.");
      return(false);
     }
//--- 레벨 숫자 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- 루프에서 레벨 속성 설정
   for(int i=0;i<levels;i++)
     {
      //--- 레벨 값
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- 레벨 색상
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- 레벨 스타일
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- 레벨 너비
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- 레벨 설명
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 되돌림 고정점 이동                                      |
//+------------------------------------------------------------------+
bool FiboLevelsPointChange(const long   chart_ID=0,        // 차트의 ID
                           const string name="FiboLevels", // 개체 이름
                           const int    point_index=0,     // 고정점 인덱스
                           datetime     time=0,            // 고정점 시간 좌표
                           double       price=0)           // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 되돌림 삭제                                             |
//+------------------------------------------------------------------+
bool FiboLevelsDelete(const long   chart_ID=0,        // 차트의 ID
                      const string name="FiboLevels") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"피보나치 되돌림₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 되돌림 고정점 값 확인 및  |
//| 빈 값에 기본 값 설정                                             |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- 두 번재 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 두 번째 점의 가격이 설정되지 않은 경우 입찰 값이 지정됩니다
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 첫 번째 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9 막대가 위치합니다
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 점의 가격이 설정되지 않은 경우 두 번째 점보다 200 포인트 낮게 이동합니다
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 피보나치 되돌림 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("시간 값 복사 실패! Error code = ",GetLastError());
       return;
      }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 피보나치 되돌림을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 개체 생성
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
      {
       return;
      }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy*2/5;
//--- 첫 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
      {
       //--- 다음의 값 사용
       if(p1>1)
          p1-=1;
       //--- 점 이동
       if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
          return;
       //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
       if(IsStopped())
          return;
       //--- 차트 다시 그리기
       ChartRedraw();
      }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy*4/5;
//--- 두 번째 고정점 이동
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p2>1)
         p2-=1;
      //--- 점 이동
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 개체 삭제
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOTIMES

피보나치 타임존.

참조

"피보나치 타임존"의 경우 선 레벨 수, 선 레벨 값 및 색상을 지정할 수 있습니다.

예

다음 스크립트는 차트에서 피보나치 타임존을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩피보나치 타임존₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 다음 크기의 백분율로 설정됩니다"
#property description "차트 창."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string           InpName="FiboTimes";       // 개체 이름
input int              InpDate1=10;               // 첫 번째 점 날짜(%)
input int              InpPrice1=45;              // 첫 번째 점 가격(%)
input int              InpDate2=20;               // 두 번째 점 날짜(%)
input int              InpPrice2=55;              // 두 번째 점 가격(%)
input color            InpColor=clrRed;           // 개체 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 선 스타일
input int              InpWidth=2;                // 선 너비
input bool             InpBack=false;             // 배경에
```

© 2000-2025, MetaQuotes Ltd.

```
input bool             InpSelection=true;          // 이동하려면 강조 표시
input bool             InpHidden=true;             // 개체 목록에 숨겨짐
input long             InpZOrder=0;                // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 피보나치 타임존 생성                      |
//+------------------------------------------------------------------+
bool FiboTimesCreate(const long             chart_ID=0,        // 차트의 ID
                     const string           name="FiboTimes",  // 개체 이름
                     const int              sub_window=0,      // 하위 창 인덱스
                     datetime               time1=0,           // 첫 번째 점 시간
                     double                 price1=0,          // 첫 번째 점 가격
                     datetime               time2=0,           // 두 번째 점 시간
                     double                 price2=0,          // 두 번째 점 가격
                     const color            clr=clrRed,        // 개체 색상
                     const ENUM_LINE_STYLE  style=STYLE_SOLID, // 개체 선 스타일
                     const int              width=1,           // 개체 선 너비
                     const bool             back=false,        // 배경에
                     const bool             selection=true,    // 이동하려면 강조 표시
                     const bool             hidden=true,       // 개체 목록에 숨겨짐
                     const long             z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 피보나치 타임존 생성
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": \"피보나치 타임존₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 레벨 숫자 및 레벨 매개변수 설정                                  |
//+------------------------------------------------------------------+
bool FiboTimesLevelsSet(int             levels,           // 레벨 선 수
                        double          &values[],        // 레벨 선 값
                        color           &colors[],        // 레벨 선 색상
                        ENUM_LINE_STYLE &styles[],        // 레벨 선 스타일
                        int             &widths[],        // 레벨 선 너비
                        const long      chart_ID=0,       // 차트의 ID
                        const string    name="FiboTimes") // 개체 이름
  {
//--- 배열 크기 확인
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": 오류! 배열 길이가 레벨 숫자와 일치하지 않습니다.");
      return(false);
     }
//--- 레벨 숫자 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- 루프에서 레벨 속성 설정
   for(int i=0;i<levels;i++)
     {
      //--- 레벨 값
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- 레벨 색상
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- 레벨 스타일
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- 레벨 너비
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- 레벨 설명
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 타임존 고정점 이동                                      |
//+------------------------------------------------------------------+
bool FiboTimesPointChange(const long   chart_ID=0,       // 차트의 ID
                          const string name="FiboTimes", // 개체 이름
                          const int    point_index=0,    // 고정점 인덱스
                          datetime     time=0,           // 고정점 시간 좌표
                          double       price=0)          // 고정점 가격 좌표
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 타임존 삭제                                             |
//+------------------------------------------------------------------+
bool FiboTimesDelete(const long   chart_ID=0,       // 차트의 ID
                     const string name="FiboTimes") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"피보나치 타임존₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 타임존 값 확인 및                                       |
//| 빈 값에 기본값을 설정                                            |
//+------------------------------------------------------------------+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- 첫 번째 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time1)
      time1=TimeCurrent();
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 두 번째 점의 시간이 설정되지 않은 경우 두 번째 지점에서 남은 2 막대가 위치합니다
   if(!time2)
     {
      //--- 마지막 3개 막대의 열린 시간을 수신하기 위한 배열
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- 두 번째 점에서 남은 2 막대를 첫 번째 점 설정
      time2=temp[0];
     }
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점의 가격과 동일합니다
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 피보나치 타임존 고정점의 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 피보나치 타임존을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 개체 생성
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동합니다
//--- 루프 카운터
   int h_steps=bars*2/5;
//--- 두 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d2<bars-1)
         d2+=1;
      //--- 점 이동
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   h_steps=bars*3/5;
//--- 첫 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d1<bars-1)
         d1+=1;
      //--- 점 이동
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 개체 삭제
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOFAN

피보나치 팬.

참조

"피보나치 팬"의 경우 선 레벨 수, 선 레벨 값 및 색상을 지정할 수 있습니다.

예

다음 스크립트는 차트에서 피보나치 팬을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩피보나치 팬₩ 그래픽 개체를 생성합니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="FiboFan";          // 팬 이름
input int             InpDate1=10;                // 첫 번째 점 날짜(%)
input int             InpPrice1=25;               // 첫 번째 점 가격(%)
input int             InpDate2=30;                // 두 번째 점의 날짜(%)
input int             InpPrice2=50;               // 두 번째 점 가격(%)
input color           InpColor=clrRed;            // 팬 선 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 선 스타일
input int             InpWidth=2;                 // 선 너비
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpBack=false;            // 배경에
input bool            InpSelection=true;        // 이동하려면 강조 표시
input bool            InpHidden=true;           // 개체 목록에 숨겨짐
input long            InpZOrder=0;              // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 피보나치 팬 생성                          |
//+------------------------------------------------------------------+
bool FiboFanCreate(const long            chart_ID=0,        // 차트의 ID
                   const string          name="FiboFan",    // 팬 이름
                   const int             sub_window=0,      // 하위 창 인덱스
                   datetime              time1=0,           // 첫 번째 점 시간
                   double                price1=0,          // 첫 번째 점 가격
                   datetime              time2=0,           // 두 번째 점 시간
                   double                price2=0,          // 두 번째 점 가격
                   const color           clr=clrRed,        // 팬 선 색상
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // 팬 선 스타일
                   const int             width=1,           // fan line width
                   const bool            back=false,        // 배경에
                   const bool            selection=true,    // 이동하려면 강조 표시
                   const bool            hidden=true,       // 개체 목록에 숨겨짐
                   const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 피보나치 팬 생성
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": \"피보나치 팬₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 팬을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 레벨 숫자 및 레벨 매개변수 설정                                  |
//+------------------------------------------------------------------+
bool FiboFanLevelsSet(int             levels,         // number of level lines
                      double          &values[],      // 레벨 선 값
                      color           &colors[],      // 레벨 선 색상
                      ENUM_LINE_STYLE &styles[],      // 레벨 선 스타일
                      int             &widths[],      // 레벨 선 너비
                      const long      chart_ID=0,     // 차트의 ID
                      const string    name="FiboFan") // 팬 이름
  {
//--- 배열 크기 확인
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": 오류! 배열 길이가 레벨 숫자와 일치하지 않습니다.");
      return(false);
     }
//--- 레벨 숫자 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- 루프에서 레벨 속성 설정
   for(int i=0;i<levels;i++)
     {
      //--- 레벨 값
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- 레벨 색상
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- 레벨 스타일
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- 레벨 너비
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- 레벨 설명
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 팬 고정점 이동                                          |
//+------------------------------------------------------------------+
bool FiboFanPointChange(const long   chart_ID=0,     // 차트의 ID
                        const string name="FiboFan", // 팬 이름
                        const int    point_index=0,  // 고정점 인덱스
                        datetime     time=0,         // 고정점 시간 좌표
```

© 2000-2025, MetaQuotes Ltd.

```
                        double       price=0)        // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 팬 삭제                                                 |
//+------------------------------------------------------------------+
bool FiboFanDelete(const long   chart_ID=0,     // 차트의 ID
                   const string name="FiboFan") // 팬 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 팬 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"피보나치 팬₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 팬 고정점 값 확인 및          |
//| 빈 값에 기본 값 설정                                     |
//+------------------------------------------------------------------+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- 두 번재 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 두 번째 점의 가격이 설정되지 않은 경우 입찰 값이 지정됩니다
   if(!price2)
```

© 2000-2025, MetaQuotes Ltd.

```
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 첫 번째 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9 막대가 위치합니다
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 점의 가격이 설정되지 않은 경우 두 번째 점보다 200 포인트 낮게 이동합니다
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 피보나치 팬 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
```

© 2000-2025, MetaQuotes Ltd.

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 피보나치 팬을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 개체 생성
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- now, move the fan's anchor points
//--- 루프 카운터
   int v_steps=accuracy/2;
//--- 첫 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1<accuracy-1)
         p1+=1;
      //--- 점 이동
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int h_steps=bars/4;
//--- 두 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d2<bars-1)
         d2+=1;
      //--- 점 이동
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
```

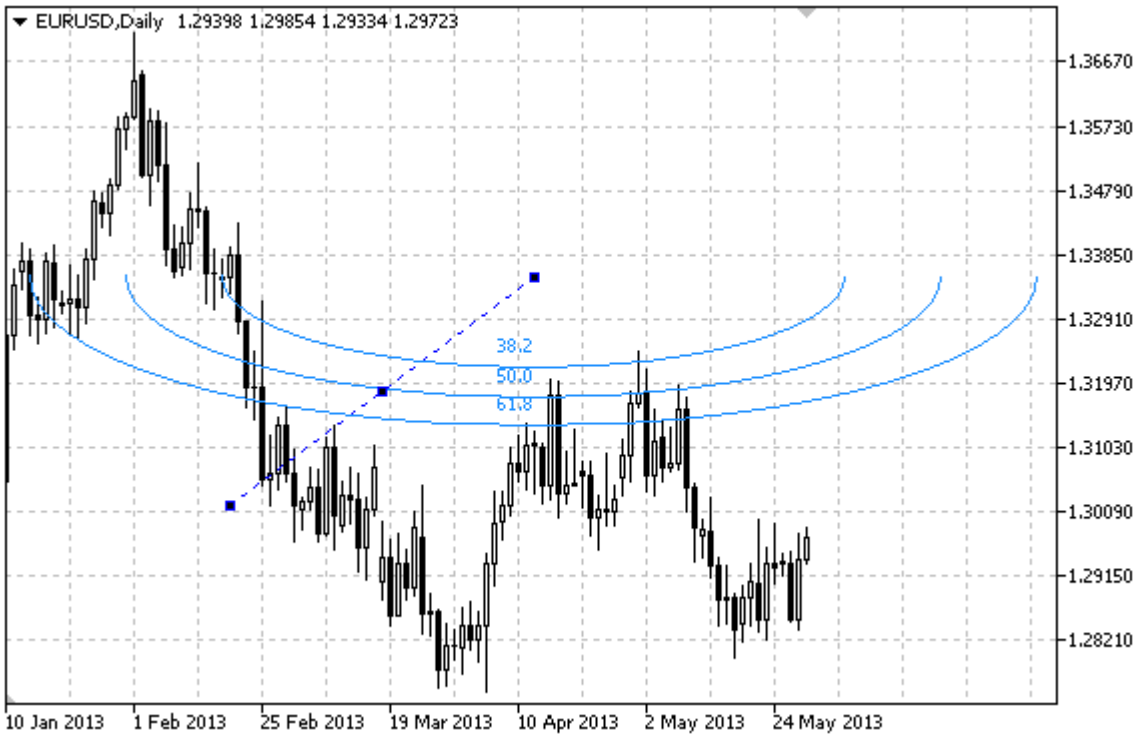
© 2000-2025, MetaQuotes Ltd.

```
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 개체 삭제
   FiboFanDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOARC

피보나치 아크.



참조

"피보나치 아크"의 경우 전체 타원의 표시 모드를 지정할 수 없습니다. 고정점의 스케일 및 좌표를 변경하여 곡률 반지름을 지정할 수 있습니다.

선 레벨 숫자, 선 수준 값 및 색상을 지정할 수도 있습니다.

예

다음 스크립트는 차트에서 피보나치 아크를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩피보나치 아크₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="FiboArc";          // 개체 이름
input int             InpDate1=25;                // 첫 번째 점 날짜(%)
input int             InpPrice1=25;               // 첫 번째 점 가격(%)
input int             InpDate2=35;                // 두 번째 점 날짜(%)
input int             InpPrice2=55;               // 두 번째 점 가격(%)
input double          InpScale=3.0;               // 스케일
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpFullEllipse=true;       // 호의 형태
input color           InpColor=clrRed;           // 선 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 선 스타일
input int             InpWidth=2;                // 선 너비
input bool            InpBack=false;             // 배경에
input bool            InpSelection=true;         // 이동하려면 강조 표시
input bool            InpHidden=true;            // 개체 목록에 숨겨짐
input long            InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 피보나치 아크 생성                        |
//+------------------------------------------------------------------+
bool FiboArcCreate(const long            chart_ID=0,          // 차트의 ID
                   const string          name="FiboArc",      // 개체 이름
                   const int             sub_window=0,        // 하위 창 인덱스
                   datetime              time1=0,             // 첫 번째 점 시간
                   double                price1=0,            // 첫 번째 점 가격
                   datetime              time2=0,             // 두 번째 점 시간
                   double                price2=0,            // 두 번째 점 가격
                   const double          scale=1.0,           // 스케일
                   const bool            full_ellipse=false,  // 호의 형태
                   const color           clr=clrRed,          // 선 색상
                   const ENUM_LINE_STYLE style=STYLE_SOLID,   // 선 스타일
                   const int             width=1,             // 선 너비
                   const bool            back=false,          // 배경에
                   const bool            selection=true,      // 이동하려면 강조 표시
                   const bool            hidden=true,         // 개체 목록에 숨겨짐
                   const long            z_order=0)           // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 피보나치 아크 생성
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": \"피보나치 아크₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 스케일 설정
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- 호 표시를 전체 타원(true) 또는 절반(false)으로 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 굵기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 호 강조 표시 모드 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 레벨 숫자 및 레벨 매개변수 설정                                  |
//+------------------------------------------------------------------+
bool FiboArcLevelsSet(int              levels,          레벨 선 숫자
                      double           &values[],       // 레벨 선 값
                      color            &colors[],       // 레벨 선 색상
                      ENUM_LINE_STYLE  &styles[],       // 레벨 선 스타일
                      int              &widths[],       // 레벨 선 너비
                      const long       chart_ID=0,      // 차트의 ID
                      const string     name="FiboArc") // 개체 이름
  {
//--- 배열 크기 확인
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": 오류! 배열 길이가 레벨 숫자와 일치하지 않습니다.");
      return(false);
     }
//--- 레벨 숫자 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- 루프에서 레벨 속성 설정
   for(int i=0;i<levels;i++)
     {
      //--- 레벨 값
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- 레벨 색상
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- 레벨 스타일
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- 레벨 너비
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- 레벨 설명
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 아크 고정점 이동                                        |
//+------------------------------------------------------------------+
bool FiboArcPointChange(const long   chart_ID=0,     // 차트의 ID
                        const string name="FiboArc", // 개체 이름
                        const int    point_index=0,  // 고정점 인덱스
                        datetime     time=0,         // 고정점 시간 좌표
                        double       price=0)        // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 아크 삭제                                               |
//+------------------------------------------------------------------+
bool FiboArcDelete(const long   chart_ID=0,     // 차트의 ID
                   const string name="FiboArc") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"피보나치 아크₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 아그 고정점 값 확인 및 기본값 설정 |
```

© 2000-2025, MetaQuotes Ltd.

```
//| 빈 곳의 값                                                        |
//+------------------------------------------------------------------+
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- 두 번재 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 두 번째 점의 가격이 설정되지 않은 경우 입찰 값이 지정됩니다
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 첫 번째 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9 막대가 위치합니다
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 점의 가격이 설정되지 않은 경우 두 번째 점보다 300 포인트 아래로 이동합니다
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 피보나치 아크 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 피보나치 아크를 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 개체 생성
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/5;
//--- 첫 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1<accuracy-1)
         p1+=1;
      //--- 점 이동
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int h_steps=bars/5;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 두 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d2<bars-1)
         d2+=1;
      //--- 점 이동
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 개체 삭제
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_FIBOCHANNEL

피보나치 채널.

참조

피보나치 채널의 경우 우측 및/또는 좌측으로 디스플레이의 연속 모드 (OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다.

선 레벨 숫자, 선 수준 값 및 색상을 지정할 수도 있습니다.

예

다음 스크립트는 차트에서 피보나치 채널을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩피보나치 채널₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="FiboChannel";      // 채널 이름
input int             InpDate1=20;                // 첫 번째 점 날짜(%)
input int             InpPrice1=10;               // 첫 번째 점 가격(%)
input int             InpDate2=60;                // 두 번째 점 날짜(%)
input int             InpPrice2=30;               // 두 번째 점 가격(%)
input int             InpDate3=20;                // 세 번째 점 날짜(%)
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpPrice3=25;              // 세 번째 점 가격(%)
input color            InpColor=clrRed;           // 채널 색상
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // 채널 선 스타일
input int              InpWidth=2;                // 채널 선 너비
input bool             InpBack=false;             // 배경 채널
input bool             InpSelection=true;         // 이동하려면 강조 표시
input bool             InpRayLeft=false;          // 채널 좌측으로 연장
input bool             InpRayRight=false;         // 채널 우측으로 연장
input bool             InpHidden=true;            // 개체 목록에 숨겨짐
input long             InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표 값을 기준으로 피보나치 채널 생성                     |
//+------------------------------------------------------------------+
bool FiboChannelCreate(const long             chart_ID=0,         // 차트의 ID
                       const string           name="FiboChannel", // 채널 이름
                       const int              sub_window=0,       // 하위 창 인덱스
                       datetime               time1=0,            // 첫 번째 점 시간
                       double                 price1=0,           // 첫 번째 점 가격
                       datetime               time2=0,            // 두 번째 점 시간
                       double                 price2=0,           // 두 번째 점 가격
                       datetime               time3=0,            // 세 번째 점 시간
                       double                 price3=0,           // 세 번째 점 가격
                       const color            clr=clrRed,         // 채널 색상
                       const ENUM_LINE_STYLE  style=STYLE_SOLID,  // 채널 선 스타일
                       const int              width=1,            // 채널 선 너비
                       const bool             back=false,         // 배경에
                       const bool             selection=true,     // 이동하려면 강조 표시
                       const bool             ray_left=false,     // 채널 좌측으로 연장
                       const bool             ray_right=false,    // 채널 우측으로 연장
                       const bool             hidden=true,        // 개체 목록에 숨겨짐
                       const long             z_order=0)          // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표로 채널 생성
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": \"피보나치 채널₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 채널 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 채널 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 채널 선 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 채널 디스플레이의 좌측 연속 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 채널 디스플레이 우측 연속 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 레벨 숫자 및 레벨 매개변수 설정                                  |
//+------------------------------------------------------------------+
bool FiboChannelLevelsSet(int             levels,             // 레벨 선 숫자
                          double          &values[],          // 레벨 선 값
                          color           &colors[],          // 레벨 선 색상
                          ENUM_LINE_STYLE &styles[],          // 레벨 선 스타일
                          int             &widths[],          // 레벨 선 너비
                          const long      chart_ID=0,         // 차트의 ID
                          const string    name="FiboChannel") // 개체 이름
  {
//--- 배열 크기 확인
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": 오류! 배열 길이가 레벨 숫자와 일치하지 않습니다.");
      return(false);
     }
//--- 레벨 숫자 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- 루프에서 레벨 속성 설정
   for(int i=0;i<levels;i++)
     {
      //--- 레벨 값
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- 레벨 색상
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- 레벨 스타일
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- 레벨 너비
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- 레벨 설명
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 채널 고정점 이동=                                       |
//+------------------------------------------------------------------+
bool FiboChannelPointChange(const long   chart_ID=0,         // 차트의 ID
                            const string name="FiboChannel", // channel name
                            const int    point_index=0,      // 고정점 인덱스
                            datetime     time=0,             // 고정점 시간 좌표
                            double       price=0)            // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 채널 삭제                                                        |
//+------------------------------------------------------------------+
bool FiboChannelDelete(const long   chart_ID=0,         // 차트의 ID
                       const string name="FiboChannel") // 채널 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 채널 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"피보나치 채널₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 채널 고정점 값 확인 및       |
//| 빈 값에 기본 값 설정                                  |
//+------------------------------------------------------------------+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- 두 번째(우측) 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time2)
      time2=TimeCurrent();
//--- 두 번째 점의 가격이 설정되지 않은 경우 입찰 값이 지정됩니다
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 첫 번째(남은) 점의 시간이 설정되지 않은 경우 두 번째 점에 남은 9개 막대에 위치합니다.
   if(!time1)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 포인트의 가격이 설정되지 않은 경우 두 번째 포인트보다 300 포인트 높게 이동합니다
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 세 번째 점의 시간이 설정되지 않은 경우 첫 번째 점의 시간과 일치합니다
   if(!time3)
      time3=time1;
//--- 세 번째 점의 가격이 설정되지 않은 경우 두 번째 점의 가격과 같습니다
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 채널 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 채널을 그리기 위한 점 정의
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 피보나치 채널 생성
   if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 채널의 고정점을 이동합니다
//--- 루프 카운터
   int h_steps=bars/10;
//--- 첫 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d1>1)
         d1-=1;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 점 이동
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int v_steps=accuracy/10;
//--- 두 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p2>1)
         p2-=1;
      //--- 점 이동
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy/15;
//--- 세 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p3<accuracy-1)
         p3+=1;
      //--- 점 이동
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 채널을 삭제
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EXPANSION

피보나치 확장.

참조

"피보나치 확장"의 경우 우측 및/또는 좌측으로 디스플레이의 연속 모드(OBJPROP_RAY_RIGHT 및 OBJPROP_RAY_LEFT 속성)를 지정할 수 있습니다.

선 레벨 숫자, 선 수준 값 및 색상을 지정할 수도 있습니다.

예

다음 스크립트는 차트에서 피보나치 확장을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩피보나치 확장₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="FiboExpansion";   // 개체 이름
input int             InpDate1=10;               // 첫 번째 점 날짜(%)
input int             InpPrice1=55;              // 첫 번째 점의 가격(%)
input int             InpDate2=30;               // 두 번째 점의 날짜(%)
input int             InpPrice2=10;              // 두 번째 점 가격(%)
input int             InpDate3=80;               // 세 번째 점 날짜(%)
```

© 2000-2025, MetaQuotes Ltd.

```
input int              InpPrice3=75;              // 세 번째 점 가격(%)
input color            InpColor=clrRed;           // 개체 색상
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // 선 스타일
input int              InpWidth=2;                // 선 너비
input bool             InpBack=false;             // 배경에
input bool             InpSelection=true;         // 이동하려면 강조 표시
input bool             InpRayLeft=false;          // 개체 좌측으로 연장
input bool             InpRayRight=false;         // 개체 우측으로 연장
input bool             InpHidden=true;            // 개체 목록에 숨겨짐
input long             InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 피보나치 확장 생성                        |
//+------------------------------------------------------------------+
bool FiboExpansionCreate(const long             chart_ID=0,           // 차트의 ID
                         const string           name="FiboExpansion", // 채널 이름
                         const int              sub_window=0,         // 하위 창 인덱스
                         datetime               time1=0,              // 첫 번째 점 시간
                         double                 price1=0,             // 첫 번째 점 가격
                         datetime               time2=0,              // 두 번째 점 시간
                         double                 price2=0,             // 두 번째 점 가격
                         datetime               time3=0,              // 세 번째 점 시간
                         double                 price3=0,             // 세 번째 점 가격
                         const color            clr=clrRed,           // 개체 색상
                         const ENUM_LINE_STYLE  style=STYLE_SOLID,    // 선 스타일
                         const int              width=1,              // 선 너비
                         const bool             back=false,           // 배경에
                         const bool             selection=true,       // 이동하려면 강조
                         const bool             ray_left=false,       // 개체 좌측으로 겨
                         const bool             ray_right=false,      // 개체 우측으로 겨
                         const bool             hidden=true,          // 개체 목록에 숨겨
                         const long             z_order=0)            // 마우스 클릭 우선
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 피보나치 확장 생성
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": \"피보나치 확장₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 개체 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선의 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 시각화 모드를 왼쪽에서 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- 개체 시각화 모드를 우측에서 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 레벨 숫자 및 레벨 매개변수 설정                                  |
//+------------------------------------------------------------------+
bool FiboExpansionLevelsSet(int              levels,                // 레벨 선 숫자
                            double           &values[],             // 레벨 선 값
                            color            &colors[],             // 레벨 선 색상
                            ENUM_LINE_STYLE  &styles[],             // 레벨 선 스타일
                            int              &widths[],             // 레벨 선 너비
                            const long       chart_ID=0,            // 차트의 ID
                            const string     name="FiboExpansion") // 개체 이름
  {
//--- 배열 크기 확인
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": 오류! 배열 길이가 레벨 숫자와 일치하지 않습니다.");
      return(false);
     }
//--- 레벨 숫자 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- 루프에서 레벨 속성 설정
   for(int i=0;i<levels;i++)
     {
      //--- 레벨 값
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- 레벨 색상
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- 레벨 스타일
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- 레벨 너비
```

© 2000-2025, MetaQuotes Ltd.

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- 레벨 설명
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 확장 고정점 이동                                        |
//+------------------------------------------------------------------+
bool FiboExpansionPointChange(const long   chart_ID=0,           // 차트의 ID
                              const string name="FiboExpansion", // 개체 이름
                              const int    point_index=0,        // 고정점 인덱스
                              datetime     time=0,               // 고정점 시간 좌표
                              double       price=0)              // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 확장 삭제                                               |
//+------------------------------------------------------------------+
bool FiboExpansionDelete(const long   chart_ID=0,           // 차트의 ID
                         const string name="FiboExpansion") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"피보나치 확장Fibonacci Expansion₩ 삭제 실패! Error code = ",GetLastErr
      return(false);
     }
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 피보나치 확장 고정점 값 확인 및     |
//| 빈 값에 기본 값 설정                                  |
//+------------------------------------------------------------------+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- 세 번째(오른쪽) 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time3)
      time3=TimeCurrent();
//--- 세 번째 점의 가격이 설정되지 않은 경우 입찰 값이 지정됩니다
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 첫 번째(좌측) 점의 시간이 설정되지 않은 경우 세 번째 지점에 9 막대가 남습니다
//--- 마지막 10개 막대의 열린 시간을 수신하기 위한 배열
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //--- 두 번째 막대에서 남은 9개 막대에 첫 번째 점을 설정
      time1=temp[0];
     }
//--- 첫 번째 점의 가격이 지정되지 않은 경우 세 번째 점의 가격과 같습니다
   if(!price1)
      price1=price3;
//--- 두 번째 점의 시간이 설정되지 않은 경우 세 번째 점의 7 막대가 남습니다
   if(!time2)
      time2=temp[2];
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점보다 250 포인트 낮게 이동합니다
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
```

© 2000-2025, MetaQuotes Ltd.

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 개체 고정점의 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 피보나치 확장을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 피보나치 확장 생성
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/10;
//--- 첫 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1>1)
```

© 2000-2025, MetaQuotes Ltd.

```
         p1-=1;
      //--- 점 이동
      if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy/2;
//--- 세 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p3>1)
         p3-=1;
      //--- 점 이동
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy*4/5;
//--- 두 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p2<accuracy-1)
         p2+=1;
      //--- 점 이동
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
//--- 차트에서 개체 삭제
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIOTWAVE5

엘리엇 동인 파동.

참조

"엘리엇 동인 파동"의 경우 선으로 점을 연결하는 모드(OBJPROP_DRAWLINES 속성)를 활성화/비활성화 할 수 있을 뿐만 아니라 파동 포지션 지정 수준(ENUM_ELLIOT_WAVE_DEGREE 열거에서)을 설정할 수 있습니다.

예

다음 스크립트는 차트에서 엘리엇 동인 파동을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩엘리엇 동인 파동₩을 그립니다."
#property description "고정점 좌표는 다음 크기의 백분율로 설정됩니다"
#property description "차트 창."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string                InpName="ElliotWave5";   // 개체 이름
input int                   InpDate1=10;             // 첫 번째 점 날자(%)
input int                   InpPrice1=90;            // 첫 번째 점 가격(%)
input int                   InpDate2=20;             // 두 번째 점의 날짜(%)
input int                   InpPrice2=40;            // 두 번째 점의 가격(%)
input int                   InpDate3=30;             // 세 번째 점의 날짜(%)
```

© 2000-2025, MetaQuotes Ltd.

```
input int                      InpPrice3=60;            // 세 번째 점의 가격(%)
input int                      InpDate4=40;             // 네 번째 점의 날짜(%)
input int                      InpPrice4=10;            // 네 번째 점의 가격(%)
input int                      InpDate5=60;             // 다섯 번째 점의 날짜(%)
input int                      InpPrice5=40;            // 다섯 번째 점의 가격(%)
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // 수준
input bool                     InpDrawLines=true;       // 선 표시
input color                    InpColor=clrRed;         // 선의 색상
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // 선의 스타일
input int                      InpWidth=2;              // 선의 너비
input bool                     InpBack=false;           // 배경 개체
input bool                     InpSelection=true;       // 이동하려면 강조 표시
input bool                     InpHidden=true;          // 개체 목록에 숨겨짐
input long                     InpZOrder=0;             // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 "엘리엇 동인 파동" 생성                     |
//+------------------------------------------------------------------+
bool ElliotWave5Create(const long                  chart_ID=0,              // 차트의
                       const string                name="ElliotWave5",      // 파동
                       const int                   sub_window=0,            // 하위
                       datetime                    time1=0,                 // 첫 번
                       double                      price1=0,                // 첫 번
                       datetime                    time2=0,                 // 두 번
                       double                      price2=0,                // 두 번
                       datetime                    time3=0,                 // 세 번
                       double                      price3=0,                // 세 번
                       datetime                    time4=0,                 // 네 번
                       double                      price4=0,                // 네 번
                       datetime                    time5=0,                 // 다섯
                       double                      price5=0,                // 다섯
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // 도
                       const bool                  draw_lines=true,         // 선 표
                       const color                 clr=clrRed,              // 개체
                       const ENUM_LINE_STYLE       style=STYLE_SOLID,       // 선 스
                       const int                   width=1,                 // 선의
                       const bool                  back=false,              // 배경에
                       const bool                  selection=true,          // 이동하
                       const bool                  hidden=true,             // 개체
                       const long                  z_order=0)               // 마우스
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,ti
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 "엘리엇 동인 파동" 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": \"엘리엇 동인 파동₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 도 설정(파동 크기)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- 선 표시 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- 개체 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선의 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 엘리엇 동인 파동 고정점 이동                                     |
//+------------------------------------------------------------------+
bool ElliotWave5PointChange(const long   chart_ID=0,         // 차트의 ID
                            const string name="ElliotWave5", // object name
                            const int    point_index=0,      // 고정점 인덱스
                            datetime     time=0,             // 고정점 시간 좌표
                            double       price=0)            // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 엘리엇 동인 파동 삭제                                            |
//+------------------------------------------------------------------+
bool ElliotWave5Delete(const long   chart_ID=0,         // 차트의 ID
                       const string name="ElliotWave5") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"엘리엇 동인 파동₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 엘리엇 동인 파동의 고정점 값을 확인 및      |
//| 빈 값에 기본값을 설정                                  |
//+------------------------------------------------------------------+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3,
                                  datetime &time4,double &price4,
                                  datetime &time5,double &price5)
  {
//--- 마지막 10개 막대의 열린 시간을 수신하기 위한 배열
   datetime temp[];
   ArrayResize(temp,10);
//--- 데이터 수신
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- 현재 차트에서 한 점 값을 수신
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 첫 번째 점의 시간이 설정되지 않은 경우 마지막 막대에서 9 막대가 남습니다
   if(!time1)
      time1=temp[0];
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 점의 시간이 설정되지 않은 경우 마지막 막대에서 마지막 7 막대가 남습니다.
   if(!time2)
      time2=temp[2];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점보다 300 포인트 낮게 이동합니다
   if(!price2)
      price2=price1-300*point;
//--- 세 번째 점의 시간이 설정되지 않은 경우 마지막 막대에서 5 막대가 남습니다.
   if(!time3)
      time3=temp[4];
//--- 세 번째 점의 가격이 설정되지 않은 경우 첫 번째 점보다 250 포인트 낮게 이동합니다
   if(!price3)
      price3=price1-250*point;
//--- 네 번째 점의 시간이 설정되지 않은 경우 마지막 막대에서 3 막대가 남습니다
   if(!time4)
      time4=temp[6];
//--- 네 번째 점의 가격이 설정되지 않은 경우 첫 번째 점보다 550 포인트 낮게 이동합니다
   if(!price4)
      price4=price1-550*point;
//--- 다섯 번째 점의 시간이 설정되지 않은 경우 마지막 막대에 표시됩니다
   if(!time5)
      time5=temp[9];
//--- 다섯 번째 점의 가격이 정해지지 않은 경우 첫 번째 포인트보다 450 포인트 낮게 이동합니다
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100 ||
      InpDate4<0 || InpDate4>100 || InpPrice4<0 || InpPrice4>100 ||
      InpDate5<0 || InpDate5>100 || InpPrice5<0 || InpPrice5>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 개체 고정점의 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 엘리엇 동인 파동을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int d4=InpDate4*(bars-1)/100;
   int d5=InpDate5*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
   int p4=InpPrice4*(accuracy-1)/100;
   int p5=InpPrice5*(accuracy-1)/100;
//--- 엘리엇 동인 파동 생성
   if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      date[d4],price[p4],date[d5],price[p5],InpDegree,InpDrawLines,InpColor,InpStyle,I
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/5;
//--- 다섯 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p5<accuracy-1)
         p5+=1;
      //--- 점 이동
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy/5;
//--- 두 번째 및 세 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값을 사용
      if(p2<accuracy-1)
         p2+=1;
      if(p3>1)
         p3-=1;
      //--- 점 변경
      if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy*4/5;
//--- 첫 번째 및 네 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값을 사용
      if(p1>1)
         p1-=1;
      if(p4<accuracy-1)
         p4+=1;
      //--- 점 변경
      if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 개체 삭제
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIOTWAVE3

엘리엇 보정 파동.

참조

"엘리엇 보정 파동"의 경우 선으로 점을 연결하는 모드(OBJPROP_DRAWLINES 속성)를 활성화/비활성화 할 수 있을 뿐만 아니라 파동 포지션 지정 수준 (ENUM_ELLIOT_WAVE_DEGREE 열거에서)을 설정할 수 있습니다.

예

다음 스크립트는 차트에서 엘리엇 보정 파동을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩엘리엇 보정 파동₩ 그래픽 개체를 그립니다."
#property description "고정점 좌표는 차트 창의 백분율로 설정됩니다"
#property description "크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string                InpName="ElliotWave3";   // 개체 이름
input int                   InpDate1=10;             // 첫 번째 점 날자(%)
input int                   InpPrice1=90;            // 첫 번째 점 가격(%)
input int                   InpDate2=30;             // 두 번째 점 날짜(%)
input int                   InpPrice2=10;            // 두 번째 점 가격(%)
input int                   InpDate3=50;             // 세 번째 점 날짜(%)
```

© 2000-2025, MetaQuotes Ltd.

```
input int                      InpPrice3=40;            // 세 번째 점 가격(%)
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // 수준
input bool                     InpDrawLines=true;       // 선 표시
input color                    InpColor=clrRed;         // 선의 색상
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // 선의 스타일
input int                      InpWidth=2;              // 선의 너비
input bool                     InpBack=false;           // 배경 개체
input bool                     InpSelection=true;       // 이동하려면 강조 표시
input bool                     InpHidden=true;          // 개체 목록에 숨겨짐
input long                     InpZOrder=0;             // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 "엘리엇 보정 파동" 생성                   |
//+------------------------------------------------------------------+
bool ElliotWave3Create(const long                     chart_ID=0,              // 차트의
                       const string                   name="ElliotWave3",      // 파동
                       const int                      sub_window=0,            // 하위
                       datetime                       time1=0,                 // 첫 번
                       double                         price1=0,                // 첫 번
                       datetime                       time2=0,                 // 두 번
                       double                         price2=0,                // 두 번
                       datetime                       time3=0,                 // 세 번
                       double                         price3=0,                // 세 번
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // 도
                       const bool                     draw_lines=true,         // 선 표
                       const color                    clr=clrRed,              // 개체
                       const ENUM_LINE_STYLE          style=STYLE_SOLID,       // 선 스
                       const int                      width=1,                 // 선의
                       const bool                     back=false,              // 배경에
                       const bool                     selection=true,          // 이동하
                       const bool                     hidden=true,             // 개체
                       const long                     z_order=0)               // 마우스
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 "엘리엇 보정 파동" 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": \"엘리엇 보정 파동₩ 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 도 설정(파동 크기)
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- 선 표시 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- 개체 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선의 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 채널을 강조 표시하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 엘리엇 보정 파동의 고정점 이동                                   |
//+------------------------------------------------------------------+
bool ElliotWave3PointChange(const long   chart_ID=0,          // 차트의 ID
                            const string name="ElliotWave3", // 개체 이름
                            const int    point_index=0,      // 고정점 인덱스
                            datetime     time=0,             // 고정점 시간 좌표
                            double       price=0)            // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 경우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 엘리엇 보정 파동 삭제                                            |
//+------------------------------------------------------------------+
bool ElliotWave3Delete(const long   chart_ID=0,          // 차트의 ID
```

© 2000-2025, MetaQuotes Ltd.

```
                        const string name="ElliotWave3") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"엘리엇 보정 파동₩ 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 엘리엇 보정 파동의 고정점 값 확인       |
//| 빈 값에 대한 기본 값을 설정                                 |
//+------------------------------------------------------------------+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- 마지막 10개 막대의 열린 시간을 수신하기 위한 배열
   datetime temp[];
   ArrayResize(temp,10);
//--- 데이터 수신
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- 현재 차트에서 한 점 값을 수신
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 첫 번째 점의 시간이 설정되지 않은 경우 마지막 막대에서 9 막대가 남습니다
   if(!time1)
      time1=temp[0];
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 점의 시간이 설정되지 않은 경우 마지막 막대에서 5 막대가 남습니다
   if(!time2)
      time2=temp[4];
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점보다 300 포인트 낮게 이동합니다
   if(!price2)
      price2=price1-300*point;
//--- 세 번째 점의 시간이 설정되지 않은 경우 마지막 막대에서 1 막대가 남습니다
   if(!time3)
      time3=temp[8];
//--- 세 번째 점의 가격이 정해지지 않은 경우 첫 번째 점보다 200 포인트 낮게 이동합니다
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 개체 고정점의 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 엘리엇 보정 파동을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 엘리엇 보정 파동 생성
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
      InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidde
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/5;
//--- 세 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p3<accuracy-1)
         p3+=1;
      //--- 점 이동
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy*4/5;
//--- 첫 번째 및 두 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값을 사용
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- 점 변경
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 개체 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_RECTANGLE

직사각형.

참조

사각형의 경우 OBJPROP_FILL 속성을 사용하여 색상이 채워지는 모드를 설정할 수 있습니다.

예

다음 스크립트는 차트에서 직사각형을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 차트에 직사각형을 생성합니다."
#property description "고정점 좌표가 설정되어 있습니다"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="Rectangle"; // 직사각형 이름
input int             InpDate1=40;         // 첫 번째 점 날짜(%)
input int             InpPrice1=40;        // 첫 번째 점 가격(%)
input int             InpDate2=60;         // 두 번째 점 날짜(%)
input int             InpPrice2=60;        // 두 번째 점 가격(%)
input color           InpColor=clrRed;     // 직사각형 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // 직사각형 선 스타일
input int             InpWidth=2;          // 직사각형 선 너비
```

© 2000-2025, MetaQuotes Ltd.

```
input bool             InpFill=true;         // 직사각형 색상 채우기
input bool             InpBack=false;        // 배경 직사각형
input bool             InpSelection=true;    // 이동하려면 강조 표시
input bool             InpHidden=true;       // 개체 목록에 숨겨짐
input long             InpZOrder=0;          // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 직사각형 생성                             |
//+------------------------------------------------------------------+
bool RectangleCreate(const long             chart_ID=0,        // 차트의 ID
                     const string           name="Rectangle",  // 직사각형 이름
                     const int              sub_window=0,      // 하위 창 인덱스
                     datetime               time1=0,           // 첫 번째 점 시간
                     double                 price1=0,          // 첫 번째 점 가격
                     datetime               time2=0,           // 두 번째 점 시간
                     double                 price2=0,          // 두 번째 점 가격
                     const color            clr=clrRed,        // 직사각형 색상
                     const ENUM_LINE_STYLE  style=STYLE_SOLID, // 직사각형 선 스타일
                     const int              width=1,           // 직사각형 선 너비
                     const bool             fill=false,        // 직사각형 색상 채우기
                     const bool             back=false,        // 배경에
                     const bool             selection=true,    // 이동하려면 강조 표시
                     const bool             hidden=true,       // 개체 목록에 숨겨짐
                     const long             z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표로 직사각형을 작성합니다
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": 직사각형 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 직사각형 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 직사각형 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 직사각형 선 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 사각형 채우기 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 직사각형을 강조 표시 모드 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 직사각형 고정점 이동                                             |
//+------------------------------------------------------------------+
bool RectanglePointChange(const long   chart_ID=0,       // 차트의 ID
                          const string name="Rectangle", // 직사각형 이름
                          const int    point_index=0,    // 고정점 인덱스
                          datetime     time=0,           // 고정점 시간 좌표
                          double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 직사각형 삭제                                                    |
//+------------------------------------------------------------------+
bool RectangleDelete(const long   chart_ID=0,       // 차트의 ID
                     const string name="Rectangle") // 직사각형 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 직사각형 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 직사각형 삭제 실패! Error code = ",GetLastError());
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 직사각형 고정점 값을 확인하고 기본값 설정    |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- 첫 번째 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time1)
      time1=TimeCurrent();
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 점의 시간이 설정되지 않은 경우 두 번째 점에서 남은 9 막대가 위치합니다
   if(!time2)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 첫 번째 점에서 남은 9개 막대에 두 번째 점을 설정
      time2=temp[0];
     }
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점보다 300 포인트 낮게 이동합니다
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 직사각형 고정점 좌표 설정 및 변경
   datetime date[];
```

© 2000-2025, MetaQuotes Ltd.

```
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 직사각형을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- 직사각형 생성
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 직사각형 고정점을 이동합니다
//--- 루프 카운터
   int h_steps=bars/2;
//--- 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값을 사용
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- 점 변경
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int v_steps=accuracy/2;
//--- 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값을 사용
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- 점 변경
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 직사각형 삭제
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TRIANGLE

삼각형.

참조

삼각형의 경우 OBJPROP_FILL 속성을 사용하여 색상이 채워지는 모드를 설정할 수 있습니다.

예

다음 스크립트는 차트에서 삼각형을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 차트에 삼각형을 생성합니다."
#property description "고정점 좌표가 설정되어 있습니다"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="Triangle";        // 삼각형 이름
input int             InpDate1=25;               // 첫 번째 점 날짜(%)
input int             InpPrice1=50;              // 첫 번째 점 가격(%)
input int             InpDate2=70;               // 두 번째 점 날짜(%)
input int             InpPrice2=70;              // 두 번째 점 가격(%)
input int             InpDate3=65;               // 세 번째 점 날짜(%)
input int             InpPrice3=20;              // 세 번째 점 가격(%)
input color           InpColor=clrRed;           // 삼각형 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 삼각형 선 스타일
```

© 2000-2025, MetaQuotes Ltd.

```
input int             InpWidth=2;                // 삼각형 선 너비
input bool            InpFill=false;             // 삼각형 색상 채우기
input bool            InpBack=false;             // 배경 삼각형
input bool            InpSelection=true;         // 이동하려면 강조 표시
input bool            InpHidden=true;            // 개체 목록에 숨겨짐
input long            InpZOrder=0;               // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 삼각형 생성                                |
//+------------------------------------------------------------------+
bool TriangleCreate(const long            chart_ID=0,        // 차트의 ID
                    const string          name="Triangle",   // 삼각형 이름
                    const int             sub_window=0,      // 하위 창 인덱스
                    datetime              time1=0,           // 첫 번째 점 시간
                    double                price1=0,          // 첫 번째 점 가격
                    datetime              time2=0,           // 두 번째 점 시간
                    double                price2=0,          // 두 번째 점 가격
                    datetime              time3=0,           // 세 번째 점 시간
                    double                price3=0,          // 세 번째 점 가격
                    const color           clr=clrRed,        // 삼각형 색상
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // 삼각형 선 스타일
                    const int             width=1,           // 삼각형 선 너비
                    const bool            fill=false,        // 삼각형 색상 채우기
                    const bool            back=false,        // 배경에
                    const bool            selection=true,    // 이동하려면 강조 표시
                    const bool            hidden=true,       // 개체 목록에 숨겨짐
                    const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 삼각형 생성
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": 삼각형 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 삼각형 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 삼각형 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 삼각형 선 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 삼각형을 채우는 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 삼각형을 강조 표시 모드 활성화(true) 또는 비활성화(false)
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 삼각형 고정점 이동 point                                         |
//+------------------------------------------------------------------+
bool TrianglePointChange(const long   chart_ID=0,      // 차트의 ID
                         const string name="Triangle", // 삼각형 이름
                         const int    point_index=0,   // 고정점 인덱스
                         datetime     time=0,          // 고정점 시간 좌표
                         double       price=0)         // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 삼각형 삭제                                                      |
//+------------------------------------------------------------------+
bool TriangleDelete(const long   chart_ID=0,      // 차트의 ID
                    const string name="Triangle") // 삼각형 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 삼각형 삭제
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": 타원 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 삼각형의 고정점 값을 확인하고 기본값을 설정합니다     |
//| 빈 곳의 값                                                       |
//+------------------------------------------------------------------+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- 첫 번째 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time1)
      time1=TimeCurrent();
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 점의 시간이 설정되지 않은 경우 두 번째 점에서 남은 9 막대가 위치합니다
   if(!time2)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 첫 번째 점에서 남은 9개 막대에 두 번째 점을 설정
      time2=temp[0];
     }
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점보다 300 포인트 낮게 이동합니다
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 세 번째 점의 시간이 설정되지 않은 경우 두 번째 점의 날짜와 일치합니다
   if(!time3)
      time3=time2;
//--- 세 번째 점의 가격이 정해지지 않은 경우 첫 번째 점의 가격과 일치합니다
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("오류! 입력 매개변수의 잘못된 값!");
       return;
      }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 삼각형 고정점의 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 삼각형을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 삼각형 생성
   if(!TriangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 삼각형 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy*3/10;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 첫 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p1>1)
         p1-=1;
      //--- 점 이동
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int h_steps=bars*9/20-1;
//--- 두 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d2>1)
         d2-=1;
      //--- 점 이동
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   v_steps=accuracy/4;
//--- 세 번째 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p3<accuracy-1)
         p3+=1;
      //--- 점 이동
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 삼각형 삭제
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ELLIPSE

타원.

참조

타원의 경우 OBJPROP_FILL 속성을 사용하여 채워지는 모드를 설정할 수 있습니다.

예

다음 스크립트는 차트에서 타원을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 차트에 타원을 생성합니다."
#property description "고정점 좌표가 설정되었습니다"
#property description "차트 창 크기의 백분율."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpName="Ellipse";          // 타원 이름
input int             InpDate1=30;                // 첫 번째 점 날짜(%)
input int             InpPrice1=20;               // 첫 번째 점 가격(%)
input int             InpDate2=70;                // 두 번째 점 날짜(%)
input int             InpPrice2=80;               // 두 번째 점 가격(%)
input int             InpDate3=50;                // 세 번째 점 날짜(%)
input int             InpPrice3=60;               // 세 번째 점 가격(%)
input color           InpColor=clrRed;            // 타원 색상
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // 타원 선 스타일
input int             InpWidth=2;                 // 타원 선 너비
```

© 2000-2025, MetaQuotes Ltd.

```
input bool            InpFill=false;              // 타원 색상 채우기
input bool            InpBack=false;              // 배경 타원
input bool            InpSelection=true;          // 이동하려면 강조 표시
input bool            InpHidden=true;             // 개체 목록에 숨겨짐
input long            InpZOrder=0;                // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 주어진 좌표를 기준으로 타원 생성                                  |
//+------------------------------------------------------------------+
bool EllipseCreate(const long            chart_ID=0,        // 차트의 ID
                   const string          name="Ellipse",    // 타원 이름
                   const int             sub_window=0,      // 하위 창 인덱스
                   datetime              time1=0,           // 첫 번째 점 시간
                   double                price1=0,          // 첫 번째 점 가격
                   datetime              time2=0,           // 두 번째 점 시간
                   double                price2=0,          // 두 번째 점 가격
                   datetime              time3=0,           // 세 번째 점 시간
                   double                price3=0,          // 세 번째 점 가격
                   const color           clr=clrRed,        // 타원 색상
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // 타원 선 스타일
                   const int             width=1,           // 타원 선 너비
                   const bool            fill=false,        // 타원 색상 채우기
                   const bool            back=false,        // 배경에
                   const bool            selection=true,    // 이동하려면 강조 표시
                   const bool            hidden=true,       // 개체 목록에 숨겨짐
                   const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점이 설정되지 않은 경우 고정점의 좌표 설정
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- 오류 값 재설정
   ResetLastError();
//--- 주어진 좌표를 기준으로 타원 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": 타원 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 타원 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 타원 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 타원 선 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 타원 채우기 모드 활성화(참) 또는 비활성화(수정)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 이동할 타원 강조 표시 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 타원 고정점 이동                                                 |
//+------------------------------------------------------------------+
bool EllipsePointChange(const long   chart_ID=0,     // 차트의 ID
                        const string name="Ellipse", // 타원 이름
                        const int    point_index=0,  // 고정점 인덱스
                        datetime     time=0,         // 고정점 시간 좌표
                        double       price=0)        // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 경우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 타원 삭제                                                        |
//+------------------------------------------------------------------+
bool EllipseDelete(const long   chart_ID=0,     // 차트의 ID
                   const string name="Ellipse") // 타원의 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 타원 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": 타원 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 타원 고정점 값 확인 및 기본값 설정 |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- 첫 번째 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time1)
      time1=TimeCurrent();
//--- 첫 번째 점의 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 두 번째 점의 시간이 설정되지 않은 경우 두 번째 점에서 남은 9 막대가 위치합니다
   if(!time2)
     {
      //--- 마지막 10개 막대가 열린 시간을 수신하기 위한 배열
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- 첫 번째 점에서 남은 9개 막대에 두 번째 점을 설정
      time2=temp[0];
     }
//--- 두 번째 점의 가격이 설정되지 않은 경우 첫 번째 점보다 300 포인트 낮게 이동합니다
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- 세 번째 점의 시간이 설정되지 않은 경우 두 번째 점의 날짜와 일치합니다
   if(!time3)
      time3=time2;
//--- 세 번째 점의 가격이 정해지지 않은 경우 첫 번째 점의 가격과 일치합니다
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
      InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 타원 고정점의 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 타원을 그리기 위한 점 지정
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int d3=InpDate3*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
   int p3=InpPrice3*(accuracy-1)/100;
//--- 타원 생성
   if(!EllipseCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price
      InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 타원 고정점을 이동합니다
//--- 루프 카운터
   int v_steps=accuracy/5;
//--- 첫 번째 및 두 번째 고정점 이동
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값을 사용
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- 점 변경
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int h_steps=bars/5;
//--- 세 번째 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d3>1)
         d3-=1;
      //--- 점 이동
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 타원 삭제
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_THUMB_UP

추천 기호.

참조

기호에 상대적인 부착점 위치는 ENUM_ARROW_ANCHOR 열거에서 선택할 수 있습니다.

큰 기호(5개 이상)는 MetaEditor에서 코드를 작성할 때 적절한 OBJPROP_WIDTH 속성 값을 설정해야 생성할 수 있습니다.

예

다음 스크립트는 차트에서 추천 기호를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩추천₩ 기호를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpName="ThumbUp";     // 기호 이름
input int               InpDate=75;            // 고정점 날짜(%)
input int               InpPrice=25;           // 고정점 가격(%)
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // 앵커 유형
input color             InpColor=clrRed;       // 신호 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;    // 테두리 선 스타일
```

© 2000-2025, MetaQuotes Ltd.

```
input int                InpWidth=5;           //신호 크기
input bool               InpBack=false;        // 배경 기호
input bool               InpSelection=true;    // 이동하려면 강조 표시
input bool               InpHidden=true;       // 개체 목록에 숨겨짐
input long               InpZOrder=0;          // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 추천 기호 생성                                                   |
//+------------------------------------------------------------------+
bool ArrowThumbUpCreate(const long              chart_ID=0,           // 차트의 ID
                        const string            name="ThumbUp",       // 기호 이름
                        const int               sub_window=0,         // 하위 창 인덱스
                        datetime                time=0,               // 고정점 시간
                        double                  price=0,              // 고정점 가격
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // 앵커 유형
                        const color             clr=clrRed,           // 기호 색상
                        const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 테두리 선 스타
                        const int               width=3,              // 기호 크기
                        const bool              back=false,           // 배경에
                        const bool              selection=true,       // 이동하려면 강조
                        const bool              hidden=true,          // 개체 목록에 숨
                        const long              z_order=0)            // 마우스 클릭 우
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"추천₩ 기호 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 기호 설정 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 기호 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 기호를 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowThumbUpMove(const long   chart_ID=0,     // 차트의 ID
                      const string name="ThumbUp", // 개체 이름
                      datetime     time=0,         // 고정점 시간 좌표
                      double       price=0)        // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 추천 기호 앵커 유형 변경                                         |
//+------------------------------------------------------------------+
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // 차트의 ID
                              const string            name="ThumbUp",    // 개체 이름
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // 앵커 유형
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 앵커 유형 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": 앵커 유형 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 추천 기호 삭제                                                   |
//+------------------------------------------------------------------+
bool ArrowThumbUpDelete(const long   chart_ID=0,     // 차트의 ID
                        const string name="ThumbUp") // 기호 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": failed to delete \"추천₩ 기호 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                 |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 부호 고정점 좌표 설정 및 변경용
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 기호를 그리기 위한 점을 지정
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 추천 기호 생성
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동하고 부호를 기준으로 위치를 변경
//--- 루프 카운터
   int h_steps=bars/4;
//--- 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d>1)
         d-=1;
      //--- 점 이동
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
```

© 2000-2025, MetaQuotes Ltd.

```
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int v_steps=accuracy/4;
//--- 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p<accuracy-1)
         p+=1;
      //--- 점 이동
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 기호에 상대적인 고정점 위치 변경
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- 차트 다시 그리기
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//--- 도표에서 부호 삭제
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_THUMB_DOWN

비추천 기호.

참조

기호에 상대적인 부착점 위치는 ENUM_ARROW_ANCHOR 열거에서 선택할 수 있습니다.

큰 기호(5개 이상)는 MetaEditor에서 코드를 작성할 때 적절한 OBJPROP_WIDTH 속성 값을 설정해야 생성할 수 있습니다.

예

다음 스크립트는 차트에서 비추천 기호를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩비추천₩ 기호를 그립니다."
#property description "고정점 좌표는 백분율로 설정됩니다"
#property description "차트 창 크기."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string             InpName="ThumbDown";      // 기호 이름
input int                InpDate=25;               // 고정점 날짜(%)
input int                InpPrice=75;              // 고정점 가격(%)
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // 앵커 유형
input color              InpColor=clrRed;          // 기호 색상
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;       // 테두리 선 스타일
input int               InpWidth=5;               // 기호 크기
input bool              InpBack=false;            // 배경 기호
input bool              InpSelection=true;        // 이동하려면 강조 표시
input bool              InpHidden=true;           // 개체 목록에 숨겨짐
input long              InpZOrder=0;              // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 비추천 기호 생성                                                 |
//+------------------------------------------------------------------+
bool ArrowThumbDownCreate(const long              chart_ID=0,           // 차트의 ID
                          const string            name="ThumbDown",     // 기호 이름
                          const int               sub_window=0,         // 하위 창 인덱
                          datetime                time=0,               // 고정점 날짜
                          double                  price=0,              // 고정점 가격
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // 앵커 유형
                          const color             clr=clrRed,           // 기호 색상
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 테두리 선 스
                          const int               width=3,              // 기호 크기
                          const bool              back=false,           // 배경에
                          const bool              selection=true,       // 이동하려면 강
                          const bool              hidden=true,          // 개체 목록에
                          const long              z_order=0)            // 마우스 클릭
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"비추천₩ 기호 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 기호 설정 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 기호 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 기호를 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowThumbDownMove(const long   chart_ID=0,        // 차트의 ID
                        const string name="ThumbDown", // 개체 이름
                        datetime     time=0,           // 고정점 시간 좌표
                        double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 비추천 기호 앵커 유형 변경                                       |
//+------------------------------------------------------------------+
bool ArrowThumbDownAnchorChange(const long              chart_ID=0,        // 차트의 I
                                const string            name="ThumbDown",  // 개체 이름
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // 앵커 유형
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 앵커 유형 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": 앵커 유형 변경 실패! Error code = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 비추천 기호 삭제                                                 |
//+------------------------------------------------------------------+
bool ArrowThumbDownDelete(const long   chart_ID=0,       // 차트의 ID
                          const string name="ThumbDown") // 기호 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"비추천₩ 기호 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                                    |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 부호 고정점 좌표 설정 및 변경용
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 기호를 그리기 위한 점을 지정
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 비추천 기호 생성
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동하고 부호를 기준으로 위치를 변경
//--- 루프 카운터
   int h_steps=bars/4;
//--- 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d<bars-1)
         d+=1;
      //--- 점 이동
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프 카운터
   int v_steps=accuracy/4;
//--- 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p>1)
         p-=1;
      //--- 점 이동
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 기호에 상대적인 고정점 위치 변경
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- 차트 다시 그리기
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//--- 도표에서 부호 삭제
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_UP

위쪽 화살표 기호.

참조

기호에 상대적인 부착점 위치는 ENUM_ARROW_ANCHOR 열거에서 선택할 수 있습니다.

큰 기호(5개 이상)는 MetaEditor에서 코드를 작성할 때 적절한 OBJPROP_WIDTH 속성 값을 설정해야 생성할 수 있습니다.

예

다음 스크립트는 차트에서 위쪽 화살표 기호를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩위쪽 화살표₩ 기호를 그립니다."
#property description "고정점 좌표가 설정됨"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpName="ArrowUp";     // 기호 이름
input int               InpDate=25;            // 고정점 날짜(%)
input int               InpPrice=25;           // 고정점 가격(%)
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // 앵커 유형
input color             InpColor=clrRed;       // 신호 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;    // 테두리 선 스타일
```

© 2000-2025, MetaQuotes Ltd.

```
input int                InpWidth=5;           //신호 크기
input bool               InpBack=false;        // 배경 기호
input bool               InpSelection=false;   // 이동하려면 강조 표시
input bool               InpHidden=true;       // 개체 목록에 숨겨짐
input long               InpZOrder=0;          // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 위쪽 화살표 기호 생성                                            |
//+------------------------------------------------------------------+
bool ArrowUpCreate(const long              chart_ID=0,           // 차트의 ID
                   const string            name="ArrowUp",       // 기호 이름
                   const int               sub_window=0,         // 하위 창 인덱스
                   datetime                time=0,               // 고정점 시간
                   double                  price=0,              // 고정점 가격
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // 앵커 유형
                   const color             clr=clrRed,           // 기호 색상
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 테두리 선 스타일
                   const int               width=3,              // 기호 크기
                   const bool              back=false,           // 배경에
                   const bool              selection=true,       // 이동하려면 강조 표시
                   const bool              hidden=true,          // 개체 목록에 숨겨짐
                   const long              z_order=0)            // 마우스 클릭 우선 순
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"화살표 위쪽₩ 기호 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 기호 설정 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 기호 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 기호를 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowUpMove(const long   chart_ID=0,     // 차트의 ID
                 const string name="ArrowUp", // 개체 이름
                 datetime     time=0,         // 고정점 시간 좌표
                 double       price=0)        // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 아래쪽 화살표 기호 앵커 유형 변경                                |
//+------------------------------------------------------------------+
bool ArrowUpAnchorChange(const long              chart_ID=0,        // 차트의 ID
                         const string            name="ArrowUp",    // 개체 이름
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // 개체 유형
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 위치 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": 앵커 유형 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 위쪽 화살표 기호 삭제                                              |
//+------------------------------------------------------------------+
bool ArrowUpDelete(const long   chart_ID=0,     // 차트의 ID
                   const string name="ArrowUp") // 기호 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"위쪽 화살표₩ 기호 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                 |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 부호 고정점 좌표 설정 및 변경용
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 기호를 그리기 위한 점을 지정
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 위쪽 화살표 기호 생성
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동하고 부호를 기준으로 위치를 변경
//--- 루프 카운터
   int v_steps=accuracy/2;
//--- 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p<accuracy-1)
         p+=1;
      //--- 점 이동
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
```

© 2000-2025, MetaQuotes Ltd.

```
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 기호에 상대적인 고정점 위치 변경
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- 차트 다시 그리기
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//--- 도표에서 부호 삭제
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_DOWN

아래쪽 화살표 기호.

참조

기호에 상대적인 부착점 위치는 ENUM_ARROW_ANCHOR 열거에서 선택할 수 있습니다.

큰 기호(5개 이상)는 MetaEditor에서 코드를 작성할 때 적절한 OBJPROP_WIDTH 속성 값을 설정해야 생성할 수 있습니다.

예

다음 스크립트는 차트에서 아래쪽 화살표 기호를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩아래쪽 화살표₩ 기호를 그립니다."
#property description "고정점 좌표가 설정됨"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpName="ArrowDown";     // 신호 이름
input int               InpDate=75;              // Anchor point date in %
input int               InpPrice=75;             // 고정점 가격(%)
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // 앵커 유형
input color             InpColor=clrRed;         // 기호 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // 테두리 선 스타일
input int               InpWidth=5;              // 기호 크기
```

© 2000-2025, MetaQuotes Ltd.

```
input bool               InpBack=false;           // 배경 기호
input bool               InpSelection=false;      // 이동하려면 강조 표시
input bool               InpHidden=true;          // 개체 목록에 숨겨짐
input long               InpZOrder=0;             // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 아래쪽 화살표 기호 생성                                          |
//+------------------------------------------------------------------+
bool ArrowDownCreate(const long              chart_ID=0,           // 차트의 ID
                     const string            name="ArrowDown",     // 기호 이름
                     const int               sub_window=0,         // 하위 창 인덱스
                     datetime                time=0,               // 고정점 시간
                     double                  price=0,              // 고정점 가격
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // 앵커 유형
                     const color             clr=clrRed,           // 기호 색상
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 테두리 선 스타일
                     const int               width=3,              // 기호 크기
                     const bool              back=false,           // 배경에
                     const bool              selection=true,       // 이동하려면 강조 표
                     const bool              hidden=true,          // 개체 목록에 숨겨짐
                     const long              z_order=0)            // 마우스 클릭 우선 
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"아래쪽 화살표₩ 생성이 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 앵커 유형
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 기호 설정 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 기호 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 기호를 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowDownMove(const long   chart_ID=0,       // 차트의 ID
                   const string name="ArrowDown", // 개체 이름
                   datetime     time=0,           // 고정점 시간 좌표
                   double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 아래쪽 화살표 기호 앵커 유형 변경                                |
//+------------------------------------------------------------------+
bool ArrowDownAnchorChange(const long              chart_ID=0,        // 차트의 ID
                           const string            name="ArrowDown",  // 개체 이름
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // 앵커 유형
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 위치 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": 앵커 유형 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 아래쪽 화살표 기호 삭제                                          |
//+------------------------------------------------------------------+
bool ArrowDownDelete(const long   chart_ID=0,       // 차트의 ID
                     const string name="ArrowDown") // 기호 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"아래쪽 화살표₩ 기호 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                 |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 부호 고정점 좌표 설정 및 변경용
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 기호를 그리기 위한 점을 지정
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 아래쪽 화살표 기호 생성
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동하고 부호를 기준으로 위치를 변경
//--- 루프 카운터
   int v_steps=accuracy/2;
//--- 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p>1)
         p-=1;
      //--- 점 이동
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 1초 지연
   Sleep(1000);
//--- 기호에 상대적인 고정점 위치 변경
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- 차트 다시 그리기
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//--- 도표에서 부호 삭제
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_STOP

중지 기호.

참조

기호에 상대적인 부착점 위치는 ENUM_ARROW_ANCHOR 열거에서 선택할 수 있습니다.

큰 기호(5개 이상)는 MetaEditor에서 코드를 작성할 때 적절한 OBJPROP_WIDTH 속성 값을 설정해야 생성할 수 있습니다.

예

다음 스크립트는 차트에서 중지 기호를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트가 ₩중지₩ 기호를 그립니다."
#property description "고정점 좌표가 설정됨"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpName="ArrowStop";      // 기호 이름
input int               InpDate=10;               // 고정점 날짜(%)
input int               InpPrice=50;              // 고정점 가격(%)
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // 앵커 유형
input color             InpColor=clrRed;          // 기호 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;       // 테두리 선 스타일
```

© 2000-2025, MetaQuotes Ltd.

```
input int                InpWidth=5;              // 기호 크기
input bool               InpBack=false;           // 배경 기호
input bool               InpSelection=false;      // 이동하려면 강조 표시
input bool               InpHidden=true;          // 개체 목록에 숨겨짐
input long               InpZOrder=0;             // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 중지 기호 생성                                                   |
//+------------------------------------------------------------------+
bool ArrowStopCreate(const long              chart_ID=0,            // 차트의 ID
                     const string            name="ArrowStop",      // 기호 이름
                     const int               sub_window=0,          // 하위 창 인덱스
                     datetime                time=0,                // 고정점 시간
                     double                  price=0,               // 고정점 가격
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // 앵커 유형
                     const color             clr=clrRed,            // 기호 색상
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,     // 테두리 선 스타일
                     const int               width=3,               // 기호 크기
                     const bool              back=false,            // 배경에
                     const bool              selection=true,        // 이동하려면 강조 표
                     const bool              hidden=true,           // 개체 목록에 숨겨짐
                     const long              z_order=0)             // 마우스 클릭 우선 
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"중지₩ 기호 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 기호 설정 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 기호 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 기호를 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowStopMove(const long   chart_ID=0,       // 차트의 ID
                   const string name="ArrowStop", // 개체 이름
                   datetime     time=0,           // 고정점 시간 좌표
                   double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 중지 기호 앵커 유형 변경                                         |
//+------------------------------------------------------------------+
bool ArrowStopAnchorChange(const long              chart_ID=0,        // 차트의 ID
                           const string            name="ArrowStop",  // 개체 이름
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // 고정점 포지션
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 앵커 유형 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": 앵커 유형 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 중지 기호 삭제                                                   |
//+------------------------------------------------------------------+
bool ArrowStopDelete(const long   chart_ID=0,       // 차트의 ID
                     const string name="ArrowStop") // 레이블 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"중지 ₩ 기호 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                 |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 부호 고정점 좌표 설정 및 변경용
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 기호를 그리기 위한 점을 지정
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 중지 기호 생성
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동하고 부호를 기준으로 위치를 변경
//--- 루프 카운터
   int h_steps=bars*2/5;
//--- 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d<bars-1)
         d+=1;
      //--- 점 이동
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
```

© 2000-2025, MetaQuotes Ltd.

```
      ChartRedraw();
      // 0.025초 지연
      Sleep(25);
     }
//--- 기호에 상대적인 고정점 위치 변경
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- 차트 다시 그리기
   ChartRedraw();
//--- 루프 카운터
   h_steps=bars*2/5;
//--- 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d<bars-1)
         d+=1;
      //--- 점 이동
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.025초 지연
      Sleep(25);
     }
//--- 1초 지연
   Sleep(1000);
//--- 도표에서 부호 삭제
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_CHECK

체크 기호.

참조

기호에 상대적인 부착점 위치는 ENUM_ARROW_ANCHOR 열거에서 선택할 수 있습니다.

큰 기호(5개 이상)는 MetaEditor에서 코드를 작성할 때 적절한 OBJPROP_WIDTH 속성 값을 설정해야 생성할 수 있습니다.

예

다음 스크립트는 차트에서 체크 기호를 만들고 이동. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트가 ₩체크₩ 기호를 그립니다."
#property description "고정점 좌표가 설정됨"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpName="ArrowCheck"; // 신호 이름
input int               InpDate=10;           // 고정점 날짜(%)
input int               InpPrice=50;          // 고정점 가격(%)
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // 앵커 유형
input color             InpColor=clrRed;      // 신호 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;   // 테두리 선 스타일
input int               InpWidth=5;           //신호 크기
```

© 2000-2025, MetaQuotes Ltd.

```
input bool               InpBack=false;         // 배경 기호
input bool               InpSelection=false;    // 이동하려면 강조 표시
input bool               InpHidden=true;        // 개체 목록에 숨겨짐
input long               InpZOrder=0;           // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 체크 기호 생성                                                   |
//+------------------------------------------------------------------+
bool ArrowCheckCreate(const long              chart_ID=0,           // 차트의 ID
                      const string            name="ArrowCheck",    // 기호 이름
                      const int               sub_window=0,         // 하위 창 인덱스
                      datetime                time=0,               // 고정점 시간
                      double                  price=0,              // 고정점 가격
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // 앵커 유형
                      const color             clr=clrRed,           // 기호 색상
                      const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 테두리 선 스타일
                      const int               width=3,              // 기호 크기
                      const bool              back=false,           // 배경에
                      const bool              selection=true,       // 이동하려면 강조
                      const bool              hidden=true,          // 개체 목록에 숨겨
                      const long              z_order=0)            // 마우스 클릭 우선
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"체크₩ 기호 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 기호 설정 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 기호 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 기호를 이동하는 모드를 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowCheckMove(const long   chart_ID=0,        // 차트의 ID
                    const string name="ArrowCheck", // 개체 이름
                    datetime     time=0,            // 고정점 시간 좌표
                    double       price=0)           // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 체크 앵커 유형 변경                                              |
//+------------------------------------------------------------------+
bool ArrowCheckAnchorChange(const long              chart_ID=0,        // 차트의 ID
                            const string            name="ArrowCheck", // 개체 이름
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // 앵커 유형
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 앵커 유형 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": 앵커 유형 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 체크 기호 삭제                                                   |
//+------------------------------------------------------------------+
bool ArrowCheckDelete(const long   chart_ID=0,        // 차트의 ID
                      const string name="ArrowCheck") // 기호 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"체크₩ 기호 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                 |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 부호 고정점 좌표 설정 및 변경용
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 기호를 그리기 위한 점을 지정
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 체크 기호 생성
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 이동하고 부호를 기준으로 위치를 변경
//--- 루프 카운터
   int h_steps=bars*2/5;
//--- 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d<bars-1)
         d+=1;
      //--- 점 이동
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      // 0.025초 지연
      Sleep(25);
     }
//--- 기호에 상대적인 고정점 위치 변경
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- 차트 다시 그리기
   ChartRedraw();
//--- 루프 카운터
   h_steps=bars*2/5;
//--- 고정점 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 다음의 값 사용
      if(d<bars-1)
         d+=1;
      //--- 점 이동
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.025초 지연
      Sleep(25);
     }
//--- 1초 지연
   Sleep(1000);
//--- 도표에서 부호 삭제
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_LEFT_PRICE

좌측 가격 레이블

예

다음 스크립트는 차트에서 좌측 가격 레이블을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트가 차트 좌측 가격 레이블을 생성합니다."
#property description "고정점 좌표가 설정됨"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpName="LeftPrice"; // 가격 레이블 이름
input int               InpDate=100;         // 고정점 날짜(%)
input int               InpPrice=10;         // 고정점 가격(%)
input color             InpColor=clrRed;     // 가격 레이블 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // 테두리 선 스타일
input int               InpWidth=2;          // 가격 레이블 크기
input bool              InpBack=false;       // 배경 레이블
input bool              InpSelection=true;   // 이동하려면 강조 표시
input bool              InpHidden=true;      // 개체 목록에 숨겨짐
input long              InpZOrder=0;         // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 좌측 가격 레이블 생성                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // 차트의 ID
                          const string          name="LeftPrice",  // 가격 레이블 이름
                          const int             sub_window=0,      // 하위 창 인덱스
                          datetime              time=0,            // 고정점 시간
                          double                price=0,           // 고정점 가격
                          const color           clr=clrRed,        // 가격 레이블 색상
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // 테두리 선 스타일
                          const int             width=1,           // 가격 레이블 크기
                          const bool            back=false,        // 배경에
                          const bool            selection=true,    // 이동하려면 강조 표
                          const bool            hidden=true,       // 개체 목록에 숨겨짐
                          const long            z_order=0)         // 마우스 클릭 우선 
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 가격 레이블 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": 좌측 가격 레이블 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 레이블 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 레이블 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 레이블 이동 모드 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ArrowLeftPriceMove(const long   chart_ID=0,       // 차트의 ID
                        const string name="LeftPrice", // 레이블 이름
                        datetime     time=0,           // 고정점 시간 좌표
                        double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에서 좌측 가격 레이블 삭제                                   |
//+------------------------------------------------------------------+
bool ArrowLeftPriceDelete(const long   chart_ID=0,       // 차트의 ID
                          const string name="LeftPrice") // 레이블 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 레이블 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 좌측 가격 레이블 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                                    |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 레이블 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 레이블 그리기에 대한 점 정의
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 좌측 가격 레이블 생성
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 움직입니다
//--- 루프 카운터
   int v_steps=accuracy*4/5;
//--- 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p<accuracy-1)
         p+=1;
      //--- 점 이동
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 레이블 삭제
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_RIGHT_PRICE

우측 가격 레이블.

예

다음 스크립트는 차트에서 우측 가격 레이블을 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 차트에 우측 가격 레이블을 생성합니다."
#property description "고정점 좌표가 설정됨"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpName="RightPrice"; // 가격 레이블 이름
input int               InpDate=0;            // 고정점 날짜(%)
input int               InpPrice=90;          // 고정점 가격(%)
input color             InpColor=clrRed;      // 가격 레이블 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // 테두리 선 스타일
input int               InpWidth=2;           // 가격 레이블 크기
input bool              InpBack=false;        // 배경 레이블
input bool              InpSelection=true;    // 이동하려면 강조 표시
input bool              InpHidden=true;       // 개체 목록에 숨겨짐
input long              InpZOrder=0;          // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 우측 가격 레이블 생성                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool ArrowRightPriceCreate(const long            chart_ID=0,        // 차트의 ID
                           const string          name="RightPrice", // 가격 레이블 이름
                           const int             sub_window=0,      // 하위 창 인덱스
                           datetime              time=0,            // 고정점 시간
                           double                price=0,           // 고정점 가격
                           const color           clr=clrRed,        // 가격 레이블 색상
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // 테두리 선 스타일
                           const int             width=1,           // 가격 레이블 크기
                           const bool            back=false,        // 배경에
                           const bool            selection=true,    // 이동하려면 강조
                           const bool            hidden=true,       // 개체 목록에 숨겨
                           const long            z_order=0)         // 마우스 클릭 우선
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 가격 레이블 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": 우측 가격 레이블 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 레이블 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 레이블 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 레이블 이동 모드 활성화(true) 또는 비활성화(false)
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool ArrowRightPriceMove(const long   chart_ID=0,        // 차트의 ID
                         const string name="RightPrice", // 레이블 이름
                         datetime     time=0,            // 고정점 시간 좌표
                         double       price=0)           // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트에서 우측 가격 레이블 삭제                                   |
//+------------------------------------------------------------------+
bool ArrowRightPriceDelete(const long   chart_ID=0,        // 차트의 ID
                           const string name="RightPrice") // 레이블 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 레이블 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 우측 가격 레이블 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                                    |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 레이블 고정점 좌표 설정 및 변경
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 레이블 그리기에 대한 점 정의
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 우측 가격 레이블 생성
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

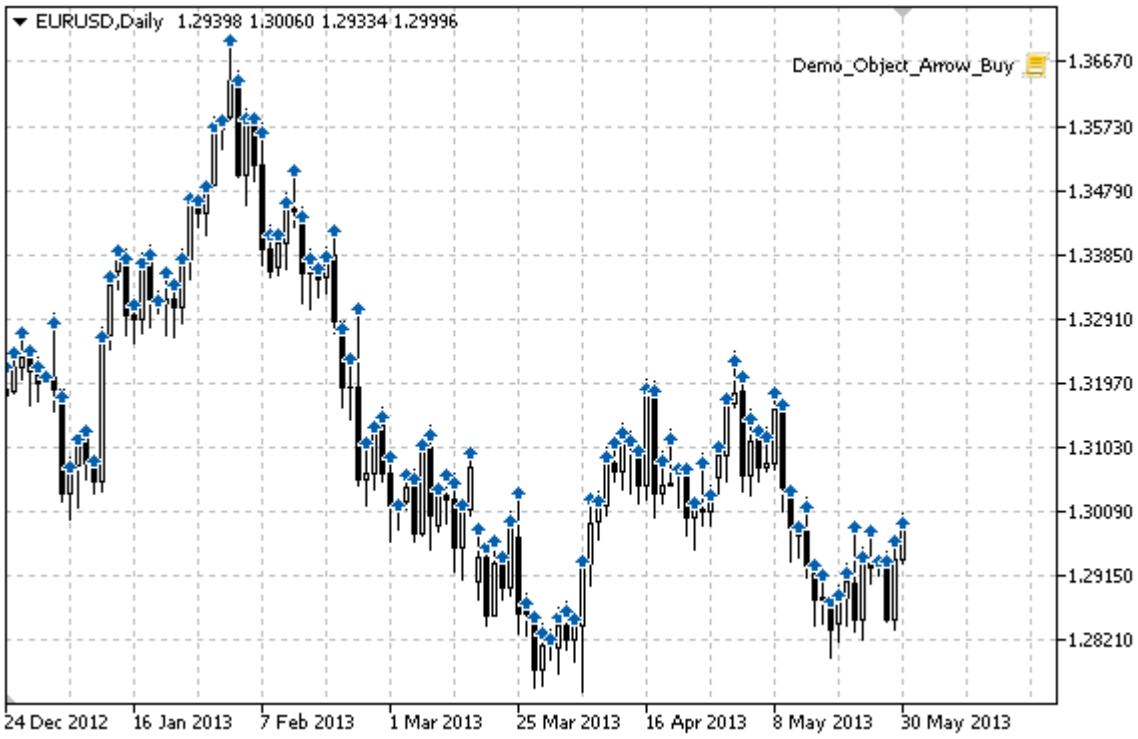
© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 고정점을 움직입니다
//--- 루프 카운터
   int v_steps=accuracy*4/5;
//--- 고정점 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 다음의 값 사용
      if(p>1)
         p-=1;
      //--- 점 이동
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 레이블 삭제
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_BUY

매수 기호.



예

다음 스크립트는 차트에서 매수 신호를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트가 차트 장에 ₩Buy\" 기호를 그립니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input color InpColor=C'3,95,172'; // 기호 색상
//+------------------------------------------------------------------+
//| 매수 기호 생성                                                   |
//+------------------------------------------------------------------+
bool ArrowBuyCreate(const long            chart_ID=0,        // 차트의 ID
                    const string          name="ArrowBuy",   // 신호 이름
                    const int             sub_window=0,      // 하위 창 인덱스
                    datetime              time=0,            // 고정점 시간
                    double                price=0,           // 고정점 가격
                    const color           clr=C'3,95,172',   // 기호 색상
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // 선 스타일(강조 표시된 경
                    const int             width=1,           // 선 크기 (강조 표시된 경우
                    const bool            back=false,        // 배경에
                    const bool            selection=false,   // 이동하려면 강조 표시
```

© 2000-2025, MetaQuotes Ltd.

```
                    const bool            hidden=true,       // 개체 목록에 숨겨짐
                    const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Buy\" 기호 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 기호 설정 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정 (강조 표시된 경우)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 크기 설정 (강조 표시된 경우)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 기호를 이동하는 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowBuyMove(const long   chart_ID=0,       // 차트의 ID
                  const string name="ArrowBuy", // 개체 이름
                  datetime     time=0,          // 고정점 시간 좌표
                  double       price=0)         // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": 고정점 이동 실패! Error code = ",GetLastError());
       return(false);
      }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 매수 기호 삭제                                                   |
//+------------------------------------------------------------------+
bool ArrowBuyDelete(const long   chart_ID=0,      // 차트의 ID
                    const string name="ArrowBuy") // 기호 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Buy\" 기호 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                  |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // 표시된 막대의 날짜를 저장하기 위한 배열
   double   low[];  // 표시된 막대의 저가 가격을 저장하기 위한 배열
   double   high[]; // 표시된 막대의 고가 가격을 저장하기 위한 배열
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 저가 가격의 배열 채우기
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("저가 가격의 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 고가 가격의 배열 채우기
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("고가 가격 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 표시된 각 막대에 대해 저가 지점에 매수 기호 생성
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 표시된 각 막대에 대해 고가 지점으로 매수 기호 이동
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 매수 기호 삭제
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW_SELL

매도 기호.

예

다음 스크립트는 차트에서 매도 기호를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트가 차트 창에 ₩매도₩ 기호를 그립니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input color InpColor=C'225,68,29'; // 기호의 색상
//+------------------------------------------------------------------+
//| 매도 기호 생성                                                   |
//+------------------------------------------------------------------+
bool ArrowSellCreate(const long            chart_ID=0,        // 차트의 ID
                     const string          name="ArrowSell",  // 기호 이름
                     const int             sub_window=0,      // 하위 창 인덱스
                     datetime              time=0,            // 고정점 시간
                     double                price=0,           // 고정점 가격
                     const color           clr=C'225,68,29',  // 기호 색상
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // 선 스타일 (강조 표시된
                     const int             width=1,           // 선 크기 (강조 표시된 경
                     const bool            back=false,        // 배경에
                     const bool            selection=false,   // 이동하려면 강조 표시
```

© 2000-2025, MetaQuotes Ltd.

```
                     const bool            hidden=true,       // 개체 목록에 숨겨짐
                     const long            z_order=0)         // 마우스 클릭 우선 순위
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"매도₩ 기호 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 기호 설정 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 선 스타일 설정 (강조 표시된 경우)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 선 크기 설정 (강조 표시된 경우)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 기호를 이동하는 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowSellMove(const long   chart_ID=0,       // 차트의 ID
                   const string name="ArrowSell", // 개체 이름
                   datetime     time=0,           // 고정점 시간 좌표
                   double       price=0)          // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print(__FUNCTION__,
             ": 고정점 이동 실패! Error code = ",GetLastError());
       return(false);
      }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 매도 기호 삭제                                                   |
//+------------------------------------------------------------------+
bool ArrowSellDelete(const long   chart_ID=0,       // 차트의 ID
                     const string name="ArrowSell") // 기호 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 기호 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"매도₩ 기호 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                  |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // 표시된 막대의 날짜를 저장하기 위한 배열
   double   low[];  // 표시된 막대의 저가 가격을 저장하기 위한 배열
   double   high[]; // 표시된 막대의 고가 가격을 저장하기 위한 배열
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 저가 가격의 배열 채우기
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("저가 가격의 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 고가 가격의 배열 채우기
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("고가 가격 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 표시된 각 막대에 대해 고가 지점에 매도 기호 생성
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 표시된 각 막대에 대해 저가 지점으로 매도 기호 이동
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 매도 기호 삭제
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_ARROW

Arrow object.

참조

개체에 상대적인 고정점 위치는 ENUM_ARROW_ANCHOR에서 확인할 수 있습니다.

큰 화살표(5개 이상)은 MetaEditor에서 코드에 쓸 때 적절한 OBJPROP_WIDTH 속성 값을 설정해야 만들 수 있습니다.

필요한 화살표 유형은 윙딩 글꼴의 기호 코드 중 하나를 설정하여 선택할 수 있습니다.

예

다음 스크립트는 차트에 화살표 개체를 생성하고 해당 유형을 변경합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 차트 창에 임의의 화살표를 생성합니다."
#property description "고정점 좌표가 설정됨"
#property description "차트 창 크기의 백분율"
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string             InpName="Arrow";        // 화살표 이름
input int                InpDate=50;             // 고정점 날짜(%)
input int                InpPrice=50;            // 고정점 가격(%)
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;   // 앵커 유형
```

© 2000-2025, MetaQuotes Ltd.

```
input color             InpColor=clrDodgerBlue; // 화살표 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;   // 테두리 선 스타일
input int               InpWidth=10;            // 화살표 크기
input bool              InpBack=false;          // 배경 화살표
input bool              InpSelection=false;     // 이동하려면 강조 표시
input bool              InpHidden=true;         // 개체 목록에 숨겨짐
input long              InpZOrder=0;            // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 화살표 생성                                                      |
//+------------------------------------------------------------------+
bool ArrowCreate(const long              chart_ID=0,           // 차트의 ID
                 const string            name="Arrow",         // 화살표 이름
                 const int               sub_window=0,         // 하위창 인덱스
                 datetime                time=0,               // 고정점 시간
                 double                  price=0,              // 고정점 가격
                 const uchar             arrow_code=252,       // 화살표 코드
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // 고정점 포지션
                 const color             clr=clrRed,           // 화살표 색상
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // 테두리 선 스타일
                 const int               width=3,              // 화살표 크기
                 const bool              back=false,           // 배경 내
                 const bool              selection=true,       // 이동하려면 강조 표시
                 const bool              hidden=true,          // 개체 목록에 숨겨짐
                 const long              z_order=0)            // 마우스 클릭 우선 순위
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeArrowEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 화살표 생성
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": 화살표 생성에 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 화살표 코드를 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 화살표 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 테두리 선 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 화살표의 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 화살표 이동 모드를 활성화(true) 또는 비활성화(false)
```

© 2000-2025, MetaQuotes Ltd.

```
//--- ObjectCreate 함수를 사용하여 그래픽 개체를 만드는 경우 개체는
//--- 기본적으로 강조 표시되고 이동됩니다. 이 메서드 내에서 선택 매개변수
//--- 기본적으로 true이므로 개체를 강조 표시하고 이동할 수 있습니다.
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool ArrowMove(const long   chart_ID=0,   // 차트의 ID
               const string name="Arrow", // 개체 이름
               datetime     time=0,       // 고정점 시간 좌표
               double       price=0)      // 고정점 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 경우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 화살표 코드 변경                                                 |
//+------------------------------------------------------------------+
bool ArrowCodeChange(const long   chart_ID=0,   // 차트의 ID
                     const string name="Arrow", // 개체 이름
                     const uchar  code=252)     // 화살표 코드
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 화살표 코드 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": 화살표 코드 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 앵커 유형 변경                                                   |
//+------------------------------------------------------------------+
bool ArrowAnchorChange(const long              chart_ID=0,        // 차트의 ID
                       const string            name="Arrow",      // 개체 이름
                       const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // 앵커 유형
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 앵커 유형 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": 앵커 유형 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 화살표 삭제                                                      |
//+------------------------------------------------------------------+
bool ArrowDelete(const long   chart_ID=0,   // 차트의 ID
                 const string name="Arrow") // 화살표 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 화살표 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 화살료 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                                    |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 가격 배열 크기
   int accuracy=1000;
//--- 사용할 날짜 및 가격 값을 저장하기 위한 배열
//--- 부호 고정점 좌표 설정 및 변경용
   datetime date[];
   double   price[];
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 가격 배열 채우기
//--- 차트의 최고값과 최저값을 찾기
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- 가격의 변경 단계를 정의 하고 배열 채우기
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- 화살표 그리기를 위한 점 지정
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- 차트에 화살표를 생성
```

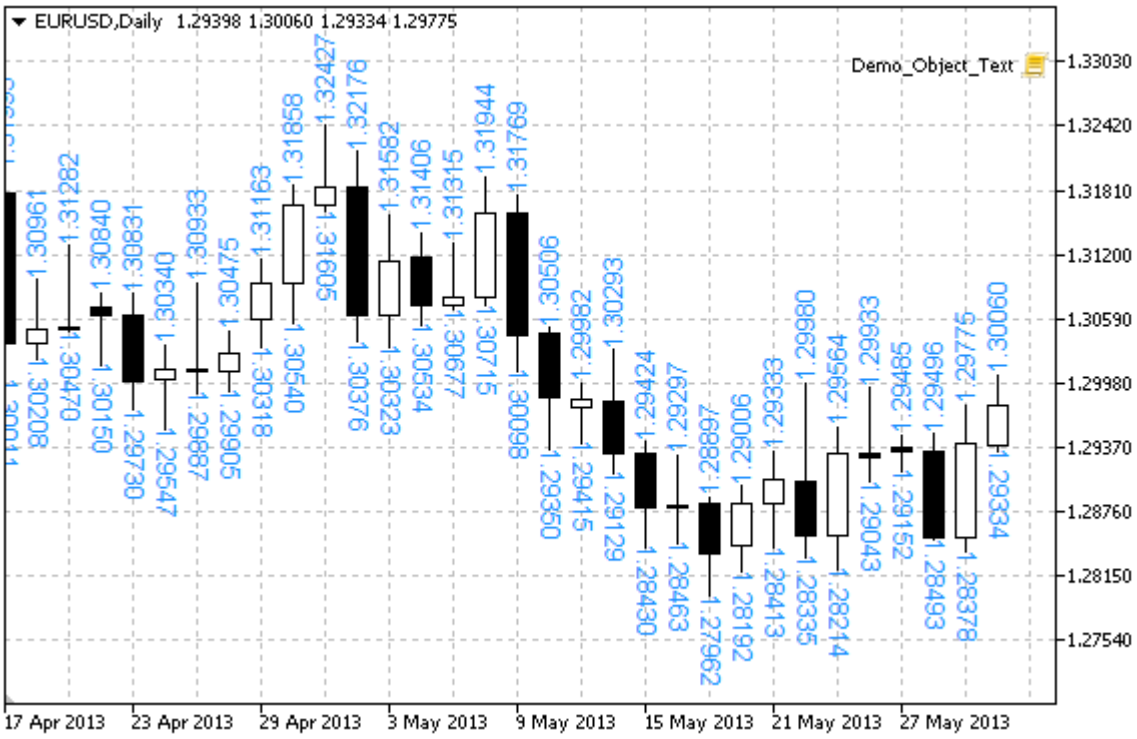
© 2000-2025, MetaQuotes Ltd.

```
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트 다시 그리기
   ChartRedraw();
//--- 루프에서 화살표를 만드는 모든 경우를 고려
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.5초 지연
      Sleep(500);
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 화살표를 삭제
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_TEXT

텍스트 개체.



참조

텍스트에 상대적인 고정점 포지션은 ENUM_ANCHOR_POINT 열거에서 선택할 수 있습니다. 텍스트 경사 각도를 OBJPROP_ANGLE 속성을 사용하여 변경할 수도 있습니다.

예

다음 스크립트는 차트에 여러 텍스트 개체를 생성합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩텍스트₩ 그래픽 개체를 생성합니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpFont="Arial";          // 글꼴
input int               InpFontSize=10;           // 글꼴 크기
input color             InpColor=clrRed;          // 색상
input double            InpAngle=90.0;            // 경사각(도)
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_LEFT;    // 앵커 유형
input bool              InpBack=false;            // 배경 개체
input bool              InpSelection=false;       // 이동하려면 강조 표시
input bool              InpHidden=true;           // 개체 목록에 숨겨짐
input long              InpZOrder=0;              // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 텍스트 개체 생성                                                  |
//+------------------------------------------------------------------+
bool TextCreate(const long              chart_ID=0,                 // 차트의 ID
                const string            name="Text",                // 개체 이름
                const int               sub_window=0,               // 하위 창 인덱스
                datetime                time=0,                     // 고정점 시간
                double                  price=0,                    // 고정점 가격
                const string            text="Text",                // 텍스트 그 자체
                const string            font="Arial",               // 글꼴
                const int               font_size=10,               // 글꼴 크기
                const color             clr=clrRed,                 // 색상
                const double            angle=0.0,                  // 텍스트 기울기
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER,   // 앵커 유형
                const bool              back=false,                 // 배경에
                const bool              selection=false,            // 이동하려면 강조 표시
                const bool              hidden=true,                // 개체 목록에 숨겨짐
                const long              z_order=0)                  // 마우스 클릭 우선 순
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeTextEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 텍스트 개체 생성
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"텍스트₩ 개체 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 텍스트 설정
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- 텍스트 글꼴 설정
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- 글꼴 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- 텍스트 경사각 설정
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- enable (true) or disable (false) the mode of moving the object by mouse
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 이동                                                      |
//+------------------------------------------------------------------+
bool TextMove(const long   chart_ID=0,  // 차트의 ID
              const string name="Text", // 개체 이름
              datetime     time=0,      // 고정점 시간 좌표
              double       price=0)     // 고정점 시간 가격 좌표
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 개체 텍스트 변경                                                 |
//+------------------------------------------------------------------+
bool TextChange(const long   chart_ID=0,  // 차트의 ID
                const string name="Text", // 개체 이름
                const string text="Text") // 텍스트
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 텍스트 변경
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": 텍스트 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 텍스트 개체 삭제                                                  |
//+------------------------------------------------------------------+
bool TextDelete(const long   chart_ID=0,  // 차트의 ID
                const string name="Text") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"텍스트₩ 개체 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                 |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // 표시된 막대의 날짜를 저장하기 위한 배열
   double   low[];  // 표시된 막대의 저가 가격을 저장하기 위한 배열
   double   high[]; // 표시된 막대의 고가 가격을 저장하기 위한 배열
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- 저가 가격의 배열 채우기
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("저가 가격의 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 고가 가격의 배열 채우기
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("고가 가격 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 텍스트 디스플레이 빈도 지정
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- 단계 지정
   int step=1;
   switch(scale)
     {
      case 0:
         step=12;
         break;
      case 1:
         step=6;
         break;
      case 2:
         step=4;
         break;
      case 3:
         step=2;
         break;
     }
//--- 고가 및 저가 막대의 값에 대한 텍스트 만들기(간격 포함)
   for(int i=0;i<bars;i+=step)
     {
      //--- 텍스트 생성
      if(!TextCreate(0,"TextHigh_"+(string)i,0,date[i],high[i],DoubleToString(high[i],
         InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
        {
         return;
        }
      if(!TextCreate(0,"TextLow_"+(string)i,0,date[i],low[i],DoubleToString(low[i],5),
         InpColor,-InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
        {
         return;
        }
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 0.5초 지연
   Sleep(500);
//--- 텍스트 삭제
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_LABEL

레이블 개체.

참조

레이블에 상대적인 고정점 위치는 ENUM_ANCHOR_POINT 열거에서 선택할 수 있습니다. 고정점 좌표는 픽셀 단위로 설정됩니다.

또한 ENUM_BASE_CORNER 열거에서 텍스트 레이블 고정 코너를 선택할 수도 있습니다.

예

다음 스크립트는 차트에서 편집 개체를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩레이블₩ 그래픽 개체를 그립니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string             InpName="Label";          // 레이블 이름
input int                InpX=150;                 // X 축 거리
input int                InpY=150;                 // Y 축 거리
input string             InpFont="Arial";          // 글꼴
input int                InpFontSize=14;           // 글꼴 크기
input color              InpColor=clrRed;          // 색상
input double             InpAngle=0.0;             // 경사각(도)
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // 앵커 유형
```

© 2000-2025, MetaQuotes Ltd.

```
input bool               InpBack=false;           // 배경 개체
input bool               InpSelection=true;       // 이동하려면 강조 표시
input bool               InpHidden=true;          // 개체 목록에 숨겨짐
input long               InpZOrder=0;             // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 텍스트 레이블 생성                                               |
//+------------------------------------------------------------------+
bool LabelCreate(const long              chart_ID=0,               // 차트의 ID
                 const string            name="Label",             // 레이블 이름
                 const int               sub_window=0,             // 하위 창 인덱스
                 const int               x=0,                      // X 좌표
                 const int               y=0,                      // Y 좌표
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // 고정하기 위한 차트
                 const string            text="Label",             // 텍스트
                 const string            font="Arial",             // 글꼴
                 const int               font_size=10,             // 글꼴 크기
                 const color             clr=clrRed,               // 색상
                 const double            angle=0.0,                // 텍스트 기울기
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // 앵커 유형
                 const bool              back=false,               // 배경에
                 const bool              selection=false,          // 이동하려면 강조 표
                 const bool              hidden=true,              // 개체 목록에 숨겨짐
                 const long              z_order=0)                // 마우스 클릭 우선 
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 텍스트 레이블 생성
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": 텍스트 레이블 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 레이블 좌표 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- 지정된 점 좌표를 기준으로 차트 모서리 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- 텍스트 설정
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- 텍스트 글꼴 설정
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- 글꼴 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- 텍스트 경사각 설정
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 색상 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 레이블 이동 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 텍스트 레이블 이동                                               |
//+------------------------------------------------------------------+
bool LabelMove(const long   chart_ID=0,   // 차트의 ID
               const string name="Label", // 레이블 이름
               const int    x=0,          // X 좌표
               const int    y=0)          // Y 좌표
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 텍스트 레이블 이동
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": 레이블의 X 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": 레이블의 Y 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 레이블 바인딩에 대한 차트 모서리 변경                            |
//+------------------------------------------------------------------+
bool LabelChangeCorner(const long             chart_ID=0,              // 차트의 ID
                       const string           name="Label",            // 레이블 이름
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // 고정을 위한
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 고정 모서리 변경
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": 고정 모서리 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 레이블 텍스트 변경                                               |
//+------------------------------------------------------------------+
bool LabelTextChange(const long   chart_ID=0,   // 차트의 ID
                     const string name="Label", // 개체 이름
                     const string text="Text")  // 텍스트
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 텍스트 변경
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": 텍스트 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 텍스트 레이블 설정                                               |
//+------------------------------------------------------------------+
bool LabelDelete(const long   chart_ID=0,   // 차트의 ID
                 const string name="Label") // 레이블 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 레이블 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 텍스트 레이블 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- 레이블 좌표를 로컬 변수에 저장
   int x=InpX;
   int y=InpY;
//--- 차트 창 크기
   long x_distance;
   long y_distance;
//--- 창 크기 설정
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("차트 너비 가져오기 실패! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("차트 높이 가져오기 실패! Error code = ",GetLastError());
      return;
     }
//--- 입력 매개변수의 정확성 확인
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 레이블 초기 텍스트를 준비
   string text;
   StringConcatenate(text,"좌측 상단 모서리: ",x,",",y);
//--- 차트에 텍스트 레이블 생성
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 0.5 초 대기
   ChartRedraw();
   Sleep(500);
//--- 레이블을 이동하는 동시에 텍스트를 변경
//--- 축별 반복 수
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- 아랫쪽으로 레이블 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 좌표 변경
      y+=2;
      //--- 레이블 이동 및 텍스트 변경
      MoveAndTextChange(x,y,"상단 좌측 모서리: ");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 0.5초 지연
   Sleep(500);
//--- 우측으로 레이블 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 좌표 변경
      x+=2;
      //--- 레이블 이동 및 텍스트 변경
      MoveAndTextChange(x,y,"상단 좌측 모서리: ");
     }
//--- 0.5초 지연
   Sleep(500);
//--- 위쪽으로 레이블 이동
   for(int i=0;i<v_steps;i++)
     {
      //--- 좌표 변경
      y-=2;
      //--- 레이블 이동 및 텍스트 변경
      MoveAndTextChange(x,y,"상단 좌측 모서리: ");
     }
//--- 0.5초 지연
   Sleep(500);
//--- 좌측으로 레이블 이동
   for(int i=0;i<h_steps;i++)
     {
      //--- 좌표 변경
      x-=2;
      //--- 레이블 이동 및 텍스트 변경
      MoveAndTextChange(x,y,"상단 좌측 모서리: ");
     }
//--- 0.5초 지연
   Sleep(500);
//--- 이제 고정 모서리를 변경하여 점을 이동합니다
//--- 좌측 하단 모서리로 이동
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- 레이블 텍스트 변경
   StringConcatenate(text,"좌측 하단 모서리: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- 차트를 다시 그리고 2 초 대기
   ChartRedraw();
   Sleep(2000);
//--- 우측 하단 모서리로 이동
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- 레이블 텍스트 변경
   StringConcatenate(text,"우측 하단 모서리: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
//--- 차트를 다시 그리고 2 초 대기
   ChartRedraw();
   Sleep(2000);
//--- 우측 상단 모서리로 이동
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- 레이블 텍스트 변경
   StringConcatenate(text,"우측 상단 모서리: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- 차트를 다시 그리고 2 초 대기
   ChartRedraw();
   Sleep(2000);
//--- 좌측 상단 모서리로 이동
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- 레이블 텍스트 변경
   StringConcatenate(text,"좌측 상단 모서리: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- 차트를 다시 그리고 2 초 대기
   ChartRedraw();
   Sleep(2000);
//--- 레이블 삭제
   LabelDelete(0,InpName);
//--- 차트를 다시 그리고 0.5 초 대기
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------+
//| 개체를 이동하고 텍스트를 변경하는 기능                           |
//+------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- 레이블 이동
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- 레이블 텍스트 변경
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return(false);
//--- 스크립트 작업이 강제로 비활성화 되었는지 확인
   if(IsStopped())
      return(false);
//--- 차트 다시 그리기
   ChartRedraw();
// 0.01 초 대기
```

© 2000-2025, MetaQuotes Ltd.

```
    Sleep(10);
//--- 기능 나가기
    return(true);
   }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BUTTON

버튼 개체.

참조

고정점 좌표는 픽셀 단위로 설정됩니다. 버튼 고정 모서리를 ENUM_BASE_CORNER에서 선택할 수 있습니다.

예

다음 스크립트는 차트에서 버튼 개체를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트가 차트에 버튼을 생성합니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string           InpName="Button";           // 버튼 이름
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // 고정하기 위한 차트 모서리
input string           InpFont="Arial";            // 글꼴
input int              InpFontSize=14;             // 글꼴 크기
input color            InpColor=clrBlack;          // 텍스트 색상
input color            InpBackColor=C'236,233,216'; // 배경 색상
input color            InpBorderColor=clrNONE;     // 테두리 색상
input bool             InpState=false;             // 누름/해제
input bool             InpBack=false;              // 배경 개체
input bool             InpSelection=false;         // 이동하려면 강조 표시
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpHidden=true;              // 개체 목록에 숨겨짐
input long              InpZOrder=0;                 // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 버튼 생성                                                        |
//+------------------------------------------------------------------+
bool ButtonCreate(const long              chart_ID=0,               // 차트의 ID
                  const string            name="Button",            // 버튼 이름
                  const int               sub_window=0,             // 하위 창 인덱스
                  const int               x=0,                      // X 좌표
                  const int               y=0,                      // Y 좌표
                  const int               width=50,                 // 버튼 너비
                  const int               height=18,                // 버튼 높이
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // 고정하기 위한 차
                  const string            text="Button",            // 텍스트
                  const string            font="Arial",             // 글꼴
                  const int               font_size=10,             // 글꼴 크기
                  const color             clr=clrBlack,             // 텍스트 색상
                  const color             back_clr=C'236,233,216',  // 배경 색상
                  const color             border_clr=clrNONE,       // 테두리 색상
                  const bool              state=false,              // 누름/해제
                  const bool              back=false,               // 배경에
                  const bool              selection=false,          // 이동하려면 강조
                  const bool              hidden=true,              // 개체 목록에 숨겨
                  const long              z_order=0)                // 마우스 클릭 우선
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 버튼 생성
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": 버튼 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 버튼 좌표 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- 버튼 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 지정된 점 좌표를 기준으로 차트 모서리 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- 텍스트 설정
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- 텍스트 글꼴 설정
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- 글꼴 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- 텍스트 색상 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 배경 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- 테두리 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 버튼 상태 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- 마우스로 버튼 이동 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 버튼 이동                                                        |
//+------------------------------------------------------------------+
bool ButtonMove(const long   chart_ID=0,    // 차트의 ID
                const string name="Button", // 버튼 이름
                const int    x=0,           // X 좌표
                const int    y=0)           // Y 좌표
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 버튼 이동
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": 버튼의 X 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": 버튼의 Y 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| Change button size                                               |
//+------------------------------------------------------------------+
bool ButtonChangeSize(const long   chart_ID=0,    // 차트의 ID
```

© 2000-2025, MetaQuotes Ltd.

```
                      const string name="Button", // 버튼 이름
                      const int    width=50,      // 버튼 너비
                      const int    height=18)     // 버튼 높이
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 버튼 크기 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": 버튼 너비 변경 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": 버튼 높이 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 버튼 바인딩에 대한 차트의 모서리 변경                            |
//+------------------------------------------------------------------+
bool ButtonChangeCorner(const long             chart_ID=0,               // 차트의 ID
                        const string           name="Button",            // 버튼 이름
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // 고정을 위한
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 고정 모서리 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": 고정 모서리 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 버튼 텍스트 변경                                                 |
//+------------------------------------------------------------------+
bool ButtonTextChange(const long   chart_ID=0,    // 차트의 ID
                      const string name="Button", // 버튼 이름
                      const string text="Text")   // 텍스트
  {
//--- 오류 값 재설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- 개체 텍스트 변경
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": 텍스트 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 버튼 삭제                                                        |
//+------------------------------------------------------------------+
bool ButtonDelete(const long   chart_ID=0,    // 차트의 ID
                  const string name="Button") // 버튼 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 버튼 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 버튼 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 차트 창 크기
   long x_distance;
   long y_distance;
//--- 창 크기 설정
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("차트 너비 가져오기 실패! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("차트 높이 가져오기 실패! Error code = ",GetLastError());
      return;
     }
//--- 버튼 크기 변경 단계 지정
```

© 2000-2025, MetaQuotes Ltd.

```
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//--- 버튼 좌표 및 크기 설정
   int x=(int)x_distance/32;
   int y=(int)y_distance/32;
   int x_size=(int)x_distance*15/16;
   int y_size=(int)y_distance*15/16;
//--- 버튼 생성
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSiz
      InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- 차트 다시 그리기
   ChartRedraw();
//--- 루프의 버튼 줄이기
   int i=0;
   while(i<13)
     {
      //--- 0.5 초 지연
      Sleep(500);
      //--- 버튼을 누름 상태로 전환
      ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
      //--- 차트를 다시 그리고 0.2초 동안 대기
      ChartRedraw();
      Sleep(200);
      //--- 좌표 및 버튼 크기 재지정
      x+=x_step;
      y+=y_step;
      x_size-=x_step*2;
      y_size-=y_step*2;
      //--- 버튼 줄이기
      ButtonMove(0,InpName,x,y);
      ButtonChangeSize(0,InpName,x_size,y_size);
      //--- 버튼을 해제된 상태로 되돌림
      ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
      //--- 차트 다시 그리기
      ChartRedraw();
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 루프 카운터를 증가
      i++;
     }
//--- 0.5초 지연
   Sleep(500);
//--- 버튼 삭제
   ButtonDelete(0,InpName);
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 1초 동안 대기
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_CHART

차트 개체.

참조

" OBJ_CHART" 유형 객체는 시각적 테스트 중에 지원되지 않습니다(표시되지 않음).

고정점 좌표는 픽셀 단위로 설정됩니다. 고정 모서리를 ENUM_BASE_CORNER 열거에서 선택할 수 있습니다.

차트 개체에 대해 기호, 주기 및 스케일을 선택할 수 있습니다. 가격 스케일 및 날짜 표시 모드도 활성화/비활성화할 수 있습니다.

예

다음 스크립트는 차트에서 차트 개체를 만드로 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩차트₩ 개체를 생성합니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string           InpName="Chart";              // 개체 이름
input string           InpSymbol="EURUSD";           // 기호
input ENUM_TIMEFRAMES  InpPeriod=PERIOD_H1;          // 주기
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // 고정 모서리
input int              InpScale=2;                   // 스케일
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpDateScale=true;           // 시간 스케일 표시
input bool              InpPriceScale=true;          // 가격 스케일 표시
input color             InpColor=clrRed;             // 강조 표시된 경우 테두리 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_DASHDOTDOT;   // 강조 표시된 경우 선 스타일
input int               InpPointWidth=1;             // 이동할 점 크기
input bool              InpBack=false;               // 배경 개체
input bool              InpSelection=true;           // 이동하려면 강조 표시
input bool              InpHidden=true;              // 개체 목록에 숨겨짐
input long              InpZOrder=0;                 // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 차트 개체 생성                                                   |
//+------------------------------------------------------------------+
bool ObjectChartCreate(const long              chart_ID=0,               // 차트의 ID
                       const string            name="Chart",             // 개체 이름
                       const int               sub_window=0,             // 하위 창 인덱
                       const string            symbol="EURUSD",          // 기호
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,         // 주기
                       const int               x=0,                      // X 좌표
                       const int               y=0,                      // Y 좌표
                       const int               width=300,                // 너비
                       const int               height=200,               // 높이
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // 고정 모서리
                       const int               scale=2,                  // 스케일
                       const bool              date_scale=true,          // 시간 스케일
                       const bool              price_scale=true,         // 가격 스케일
                       const color             clr=clrRed,               // 강조 표시된
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // 강조 표시된
                       const int               point_width=1,            // 점 크기 이동
                       const bool              back=false,               // 배경에
                       const bool              selection=false,          // 이동하려면
                       const bool              hidden=true,              // 개체 목록에
                       const long              z_order=0)                // 마우스 클릭
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 차트 개체 생성
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": \"차트₩ 개체 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 개체 좌표 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- 개체 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 지정된 점 좌표를 기준으로 차트 모서리 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- 기호 설정
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- 주기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- 스케일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- 시간 스케일 표시(true) 또는 숨김(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- 가격 스케일 표시(true) 또는 숨김(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- 개체 강조 모드가 활성화된 경우 테두리 색 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 개체 강조 모드가 활성화된 경우 테두리 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 개체를 이동할 고정점 크기를 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 레이블 이동 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트 개체의 기호 및 시간 프레임을 설정                           |
//+------------------------------------------------------------------+
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,        // 차트의 I
                                   const string          name="Chart",      // 개체 이름
                                   const string          symbol="EURUSD",   // 기호
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // 시간 프레
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 차트 개체의 기호 및 시간 프레임 설정
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            ": \"차트\" 개체의 기호 설정 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": \"차트\" 개체의 주기 설정 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트 개체 이동                                                   |
//+------------------------------------------------------------------+
bool ObjectChartMove(const long   chart_ID=0,   // 차트의 ID (차트 개체의 ID가 아닌)
                     const string name="Chart", // 개체 이름
                     const int    x=0,          // X 좌표
                     const int    y=0)          // Y 좌표
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 이동
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": \"차트₩ 개체의 X 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": \"차트₩ 개체의 Y 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트 개체 크기 변경                                              |
//+------------------------------------------------------------------+
bool ObjectChartChangeSize(const long   chart_ID=0,   // 차트의 ID (차트 개체의 ID가 아닌
                           const string name="Chart", // 개체 이름
                           const int    width=300,    // 너비
                           const int    height=200)   // 높이
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 크기 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": \"차트₩ 개체의 너비 변경 실패₩ Error code = ",GetLastError());
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": \"차트₩ 개체의 높이 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트 개체의 ID 반환                                              |
//+------------------------------------------------------------------+
long ObjectChartGetID(const long   chart_ID=0,   // 차트의 ID (차트 개체의 ID가 아닌)
                      const string name="Chart") // 개체 이름
  {
//--- 변수를 준비하여 차트 개체의 ID를 가져오기
   long id=-1;
//--- 오류 값 재설정
   ResetLastError();
//--- ID 가져오기
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            ": \"차트₩ 개체의 ID 가져오기 실패! Error code = ",GetLastError());
     }
//--- 결과 반환
   return(id);
  }
//+------------------------------------------------------------------+
//| 차트 개체 삭제                                                   |
//+------------------------------------------------------------------+
bool ObjectChartDelete(const long   chart_ID=0,   // 차트의 ID (차트 개체의 ID가 아닌)
                       const string name="Chart") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 버튼 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"차트₩ 개체 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- 마켓 워치(Market Watch)의 기호 수 가져오기
   int  symbols=SymbolsTotal(true);
//--- 지정된 이름의 기호가 기호 목록에 있는지 확인합니다
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("오류! ",InpSymbol," 기호가 ₩Market Watch\에 존재하지 않습니다"!");
      return;
     }
//--- 입력 매개변수의 유효성 검사
   if(InpScale<0 || InpScale>5)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }

//--- 차트 창 크기
   long x_distance;
   long y_distance;
//--- 창 크기 설정
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("차트 너비 가져오기 실패! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("차트 높이 가져오기 실패! Error code = ",GetLastError());
      return;
     }
//--- 차트 개체 좌표 및 해당 크기 설정
   int x=(int)x_distance/16;
   int y=(int)y_distance/16;
   int x_size=(int)x_distance*7/16;
   int y_size=(int)y_distance*7/16;
//--- 차트 개체 생성
   if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,1
      InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
```

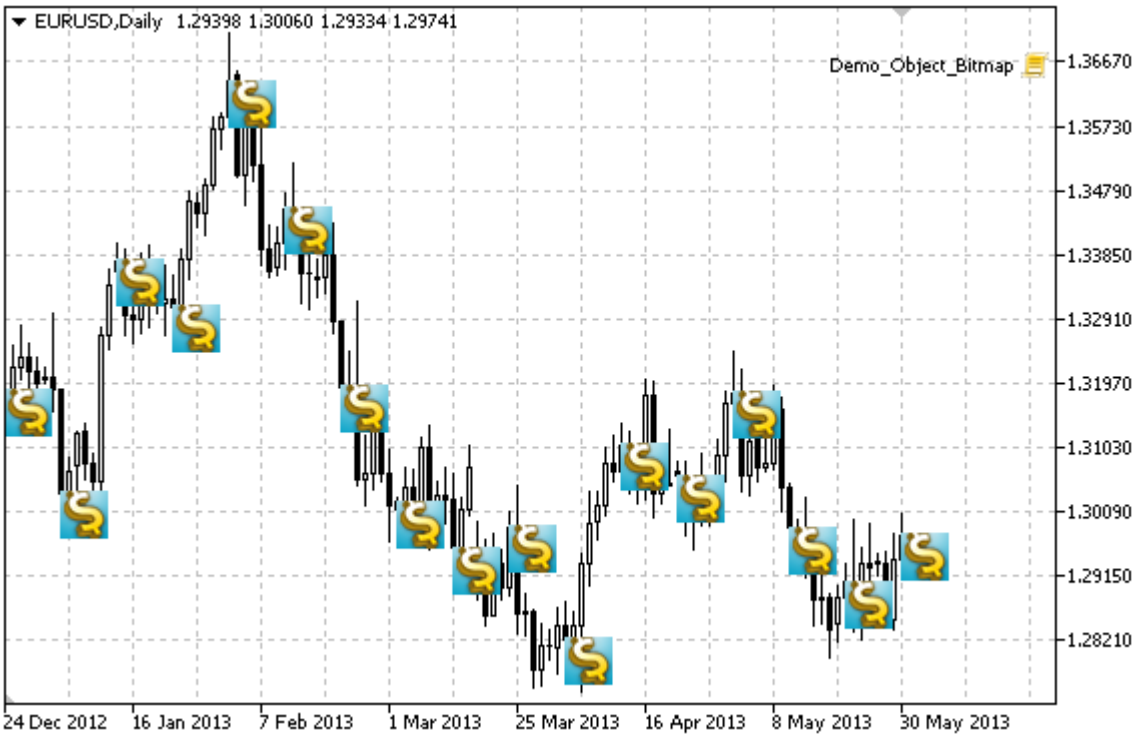
© 2000-2025, MetaQuotes Ltd.

```
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 차트 개체 늘이기
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
     {
      //--- 크기 조정
      x_size+=1;
      y_size+=1;
      if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트를 다시 그리고 0.01 초 대기
      ChartRedraw();
      Sleep(10);
     }
//--- 0.5초 지연
   Sleep(500);
//--- 차트의 시간 프레임 변경
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- 3초 지연
   Sleep(3000);
//--- 개체 삭제
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//--- 1초 동안 대기
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BITMAP

비트맵 개체.



참조

비트맵 개체의 경우 이미지의 가시성 범위를 선택할 수 있습니다.

예

다음 스크립트는 차트에서 여러 비트맵을 생성합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 차트 창에 비트맵을 생성합니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string          InpFile="\\Images\\dollar.bmp"; // 비트맵 파일 이름
input int             InpWidth=24;                    // 가시 범위 X 좌표
input int             InpHeight=24;                   // 가시 범위 Y 좌표
input int             InpXOffset=4;                   // X축 가시 범위 변경
input int             InpYOffset=4;                   // Y 축 가시 범위 변경
input color           InpColor=clrRed;                // 강조 표시된 경우 테두리 색상
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // 강조 표시된 경우 선 스타일
input int             InpPointWidth=1;                // 이동하려는 점 크기
input bool            InpBack=false;                  // 배경 개체
input bool            InpSelection=false;             // 이동하려면 강조 표시
input bool            InpHidden=true;                 // 개체 목록에 숨겨짐
```

© 2000-2025, MetaQuotes Ltd.

```
input long            InpZOrder=0;                    // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 차트 창에 비트맵 생성                                            |
//+------------------------------------------------------------------+
bool BitmapCreate(const long            chart_ID=0,         // 차트의 ID
                  const string          name="Bitmap",      // 비트맵 이름
                  const int             sub_window=0,       // 하위 창 인덱스
                  datetime              time=0,             // 고정점 시간
                  double                price=0,            // 고정점 가격
                  const string          file="",            // 비트맵 파일 이름
                  const int             width=10,           // X 좌표 가시 범위
                  const int             height=10,          // Y 좌표 가시 범위
                  const int             x_offset=0,         // X 축 가시 범위 변경
                  const int             y_offset=0,         // Y 축 가시 범위 변경
                  const color           clr=clrRed,         // 강조 표시된 경우 테두리 색상
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // 강조 표시된 경우 선 스타일
                  const int             point_width=1,      // 점 크기 이동
                  const bool            back=false,         // 배경에
                  const bool            selection=false,    // 이동하려면 강조 표시
                  const bool            hidden=true,        // 개체 목록에 숨겨짐
                  const long            z_order=0)          // 마우스 클릭 우선 순위
  {
//--- 고정점 좌표를 설정하지 않은 경우
   ChangeBitmapEmptyPoint(time,price);
//--- 오류 값 재설정
   ResetLastError();
//--- 비트맵 생성
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": 차트에 비트맵 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 이미지 파일 경로 설정
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": 이미지 불러오기 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 너비 또는 높이 값인 경우, 이미지에 대한 가시 범위 설정
//--- 원본 이미지의 너비와 높이가 (각각) 초과합니다,
//--- 반대의 경우, 그려지는 것이 아닙니다,
//--- 이 값에 해당하는 부분만 그려집니다
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 가시 범위 내에 표시할 이미지 부분을 설정
//--- 기본 부분은 이미지 좌측 상단 영역이며, 값이 허용됩니다
//--- 이 영역에서 이동 수행 이미지의 다른 부분 표시
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- 개체 강조 모드가 활성화된 경우 테두리 색 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 개체 강조 모드가 활성화된 경우 테두리 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 개체를 이동할 고정점 크기를 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 레이블 이동 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 비트맵의 새 이미지 설정                                          |
//+------------------------------------------------------------------+
bool BitmapSetImage(const long   chart_ID=0,    // 차트의 ID
                    const string name="Bitmap", // 비트맵 이름
                    const string file="")       // 파일 경로
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 이미지 파일 경로 설정
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": 이미지 불러오기 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 차트 창에 비트맵 이동                                            |
//+------------------------------------------------------------------+
bool BitmapMove(const long   chart_ID=0,    // 차트의 ID
                const string name="Bitmap", // bitmap name
                datetime     time=0,        // 고정점 시간
                double       price=0)       // 고정점 가격
  {
//--- 포인트 위치가 설정되지 않은 겨우 입찰 가격이 있는 현재 막대로 이동합니다
   if(!time)
```

© 2000-2025, MetaQuotes Ltd.

```
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- 오류 값 재설정
   ResetLastError();
//--- 고정점 이동
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": 고정점 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 가시 범위(비트맵) 크기 설정                                      |
//+------------------------------------------------------------------+
bool BitmapChangeSize(const long   chart_ID=0,    // 차트의 ID
                      const string name="Bitmap", // 비트맵 이름
                      const int    width=0,       // 비트맵 너비
                      const int    height=0)      // 비트맵 높이
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 비트맵 크기 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": 비트맵 너비 변경 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": 비트맵 높이 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 가시 범위 좌측 상단 모서리 좌표 변경 |
//+------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,    // 차트의 ID
                           const string name="Bitmap", // 비트맵 이름
                           const int    x_offset=0,    // X 좌표 가시 범위
                           const int    y_offset=0)    // Y 좌표 가시 범위
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 값 재설정
   ResetLastError();
//--- 비트맵의 가시성 범위 좌표 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": 가시 범위의 X 좌표 변경 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": 가시 범위의 Y 좌표 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 비트맵 삭제                                                      |
//+------------------------------------------------------------------+
bool BitmapDelete(const long   chart_ID=0,    // 차트의 ID
                  const string name="Bitmap") // 비트맵 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 레이블 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": 비트맵 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 고정점 값 확인 및 기본값 설정                                    |
//| 빈 곳의 경우                                                     |
//+------------------------------------------------------------------+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- 점의 시간이 설정되지 않은 경우 현재 막대에 표시됩니다
   if(!time)
      time=TimeCurrent();
//--- 포인트 가격이 설정되지 않은 경우 입찰 값을 갖습니다
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[];  // 표시된 막대의 날짜를 저장하기 위한 배열
   double   close[]; // 종가 가격을 저장하기 위한 배열
//--- 비트맵 파일 이름
   string   file="\\Images\\dollar.bmp";
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 메모리 할당
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 종가 가격 배열 채우기
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("종가 가격 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 이미지 표시 빈도 지정
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- 단계 지정
   int step=1;
   switch(scale)
     {
      case 0:
         step=27;
         break;
      case 1:
         step=14;
         break;
      case 2:
         step=7;
         break;
      case 3:
         step=4;
         break;
      case 4:
         step=2;
         break;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 고가 및 저가 막대의 값(간격 포함)에 대한 비트맵 생성
   for(int i=0;i<bars;i+=step)
     {
      //--- 비트맵 생성
      if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
         InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
        {
         return;
        }
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 0.5초 지연
   Sleep(500);
//--- 매도 기호 삭제
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_BITMAP_LABEL

비트맵 레이블 개체.

참조

레이블에 상대적인 고정점 위치는 ENUM_ANCHOR_POINT 열거에서 선택할 수 있습니다. 고정점 좌표는 픽셀 단위로 설정됩니다.

비트맵 고정 모서리를 ENUM_BASE_CORNER 열거에서 선택할 수도 있습니다.

비트맵 레이블의 경우 이미지의 가시 범위를 선택할 수 있습니다.

예

다음 스크립트는 차트에서 여러 비트맵을 생성합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩비트맵 레이블₩ 개체를 생성합니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string            InpName="BmpLabel";              // 레이블 이름
input string            InpFileOn="\\Images\\dollar.bmp"; // On 모드에 대한 파일 이름
input string            InpFileOff="\\Images\\euro.bmp";  // Off 모드에 대한 파일 이름
input bool              InpState=false;                  // 레이블 눌림/해제
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;     // 고정을 위한 차트 모서리
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER;         // 앵커 유형
input color             InpColor=clrRed;                 // 강조 표시된 경우 테두리 색상
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;            // 강조 표시된 경우 선 스타일
```

© 2000-2025, MetaQuotes Ltd.

```
input int                InpPointWidth=1;                  // 이동하려는 점 크기
input bool               InpBack=false;                    // 배경 개체
input bool               InpSelection=false;               // 이동하려면 강조 표시
input bool               InpHidden=true;                   // 개체 목록에 숨겨짐
input long               InpZOrder=0;                      // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 비트맵 레이블 개체 생성                                          |
//+------------------------------------------------------------------+
bool BitmapLabelCreate(const long              chart_ID=0,               // 차트의 ID
                       const string            name="BmpLabel",          // 레이블 이름
                       const int               sub_window=0,             // 하위 창 인덱
                       const int               x=0,                      // X 좌표
                       const int               y=0,                      // Y 좌표
                       const string            file_on="",               // On 모드의
                       const string            file_off="",              // Off 모드의
                       const int               width=0,                  // X 좌표 가시
                       const int               height=0,                 // Y 좌표의 가
                       const int               x_offset=10,              // X 축 가시
                       const int               y_offset=10,              // Y 축 가시
                       const bool              state=false,              // 누름/해제
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // 고정을 위한
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // 앵커 유형
                       const color             clr=clrRed,               // 강조 표시된
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // 강조 표시된
                       const int               point_width=1,            // 점 크기 이동
                       const bool              back=false,               // 배경에
                       const bool              selection=false,          // 이동하려면
                       const bool              hidden=true,              // 개체 목록에
                       const long              z_order=0)                // 마우스 클릭
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 비트맵 레이블 생성
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": \"비트맵 레이블₩ 개체 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- On 및 Off 모드에 대한 이미지 설정
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": On 모드 이미지 불러오기 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": Off 모드 이미지 불러오기 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 레이블 좌표 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- 너비 또는 높이 값인 경우, 이미지에 대한 가시 범위 설정
//--- 원본 이미지의 너비와 높이가 (각각) 초과합니다,
//--- 반대의 경우, 그려지는 것이 아닙니다,
//--- 이 값에 해당하는 부분만 그려집니다
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 가시 범위 내에 표시할 이미지 부분을 설정
//--- 기본 부분은 이미지 좌측 상단 영역이며, 값이 허용됩니다
//--- 이 영역에서 이동 수행 이미지의 다른 부분 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- 레이블의 상태 정의 (누름 또는 해제)
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- 지정된 점 좌표를 기준으로 차트 모서리 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- 앵커 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- 개체 강조 모드가 활성화된 경우 테두리 색 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 개체 강조 모드가 활성화된 경우 테두리 스타일 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 개체를 이동할 고정점 크기를 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 레이블 이동 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 비트맵 레이블 개체의 새 이미지 설정                              |
//+------------------------------------------------------------------+
bool BitmapLabelSetImage(const long   chart_ID=0,      // 차트의 ID
                         const string name="BmpLabel", // 레이블 이름
                         const int    on_off=0,        // 수정자 (On 또는 Off)
                         const string file="")         // 파일 경로
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 값 재설정
   ResetLastError();
//--- 이미지 파일 경로 설정
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            ": 이미지 불러오기 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 비트맵 레이블 개체 이동                                          |
//+------------------------------------------------------------------+
bool BitmapLabelMove(const long   chart_ID=0,      // 차트의 ID
                     const string name="BmpLabel", // 레이블 이름
                     const int    x=0,             // X 좌표
                     const int    y=0)             // Y 좌표
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 이동
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": 개체의 X 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": 개체의 Y 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 가시 범위(개체) 크기 변경                                        |
//+------------------------------------------------------------------+
bool BitmapLabelChangeSize(const long   chart_ID=0,      // 차트의 ID
                           const string name="BmpLabel", // 레이블 이름
                           const int    width=0,         // 레이블 너비
                           const int    height=0)        // 레이블 높이
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 크기 변경
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": 개체 너비 변경 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": 개체 높이 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 가시 범위 좌측 상단 모서리 좌표 변경 |
//+------------------------------------------------------------------+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // 차트의 ID
                                const string name="BmpLabel", // 레이블 이름
                                const int    x_offset=0,      // 가시 범위 X 좌표
                                const int    y_offset=0)      // 가시 범위 Y 좌표
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체의 가시 범위 좌표 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": 가시 범위의 X 좌표 변경 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": 가시 범위의 Y 좌표 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "비트맵 레이블" 개체 삭제                                        |
//+------------------------------------------------------------------+
bool BitmapLabelDelete(const long   chart_ID=0,      // 차트의 ID
                       const string name="BmpLabel") // 레이블 이름
  {
//--- 오류 값 재설정
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 레이블 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"비트맵 레이블₩ 개체 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 차트 창 크기
   long x_distance;
   long y_distance;
//--- 창 크기 설정
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("차트 너비 가져오기 실패! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("차트 높이 가져오기 실패! Error code = ",GetLastError());
      return;
     }
//--- 비트맵 레이블 좌표 지정
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//--- 레이블 크기 및 가시 범위 좌표 설정
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- 창 중앙에 비트맵 레이블 배치
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_
      InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidd
     {
      return;
     }
//--- 차트를 다시 그리고 1초 대기
   ChartRedraw();
   Sleep(1000);
//--- 루프에서 레이블의 가시성 범위 크기 변경
   for(int i=0;i<6;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 가시 범위 크기 변경
      width--;
      height--;
      if(!BitmapLabelChangeSize(0,InpName,width,height))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.3 초 지연
      Sleep(300);
     }
//--- 1초 지연
   Sleep(1000);
//--- 루프에서 레이블의 가시성 범위 좌표 변경
   for(int i=0;i<2;i++)
     {
      //--- 가시 범위 좌표 변경
      x_offset++;
      y_offset++;
      if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.3 초 지연
      Sleep(300);
     }
//--- 1초 지연
   Sleep(1000);
//--- 레이블 삭제
   BitmapLabelDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EDIT

개체 편집.

참조

고정점 좌표는 픽셀 단위로 설정됩니다. 고정 모서리를 ENUM_BASE_CORNER 열거에서 편집할 수 있습니다.

또한 ENUM_ALIGN_MODE 열거에서 텍스트 정렬 유형 내부 편집 중 하나를 선택할 수도 있습니다.

예

다음 스크립트는 차트에서 편집 개체를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩편집₩ 개체를 생성합니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string           InpName="Edit";              // 개체 이름
input string           InpText="Text";              // 개체 텍스트
input string           InpFont="Arial";             // 글꼴
input int              InpFontSize=14;              // 글꼴 크기
input ENUM_ALIGN_MODE  InpAlign=ALIGN_CENTER;       // 텍스트 정렬 유형
input bool             InpReadOnly=false;           // 편집 허용=
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // 고정하기 위한 차트 모서리
input color            InpColor=clrBlack;           // 텍스트 색상
input color            InpBackColor=clrWhite;       // 배경 색상
```

© 2000-2025, MetaQuotes Ltd.

```
input color            InpBorderColor=clrBlack;     // 테두리 색상
input bool             InpBack=false;               // 배경 개체
input bool             InpSelection=false;          // 이동하려면 강조 표시
input bool             InpHidden=true;              // 개체 목록에 숨겨짐
input long             InpZOrder=0;                 // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 편집 개체 생성                                                   |
//+------------------------------------------------------------------+
bool EditCreate(const long             chart_ID=0,               // 차트의 ID
                const string           name="Edit",              // 개체 이름
                const int              sub_window=0,             // 하위 창 인덱스
                const int              x=0,                      // X 좌표
                const int              y=0,                      // Y 좌표
                const int              width=50,                 // 너비
                const int              height=18,                // 높이
                const string           text="Text",              // 텍스트
                const string           font="Arial",             // 글꼴
                const int              font_size=10,             // 글꼴 크기
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,       // 정렬 유형
                const bool             read_only=false,          // 편집 능력
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // 고정을 위한 차트 모서
                const color            clr=clrBlack,             // 텍스트 색상
                const color            back_clr=clrWhite,        // background color
                const color            border_clr=clrNONE,       // 테두리 색상
                const bool             back=false,               // 배경에
                const bool             selection=false,          // 이동하려면 강조 표시
                const bool             hidden=true,              // 개체 목록에 숨겨짐
                const long             z_order=0)                // 마우스 클릭 우선 순위
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 편집 필드 생성
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": \"편집₩ 개체 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 개체 좌표 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- 개체 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 텍스트 설정
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- 텍스트 글꼴 설정
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- 글꼴 크기 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- 개체의 텍스트 정렬 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- 읽기 전용 모드의 활성화(true) 또는 비활성화(false) read-only mode
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- 지정된 개체 좌표를 기준으로 차트의 모서리 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- 텍스트 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 배경 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- 테두리 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 레이블 이동 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 편집 개체 이동                                                   |
//+------------------------------------------------------------------+
bool EditMove(const long   chart_ID=0,  // 차트의 ID
              const string name="Edit", // 개체 이름
              const int    x=0,         // X 좌표
              const int    y=0)         // Y 좌표
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 이동
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": 개체의 X 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": 개체의 Y 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 편집 개체 크기 조정                                              |
//+------------------------------------------------------------------+
bool EditChangeSize(const long   chart_ID=0,  // 차트의 ID
                    const string name="Edit", // 개체 이름
                    const int    width=0,     // 너비
                    const int    height=0)    // 높이
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 크기 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": 개체 너비 변경 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": 개체 높이 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 편집 개체의 텍스트 변경                                          |
//+------------------------------------------------------------------+
bool EditTextChange(const long   chart_ID=0,  // 차트의 ID
                    const string name="Edit", // 개체 이름
                    const string text="Text") // 텍스트
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 텍스트 변경
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": 텍스트 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 편집 개체 텍스트 반환                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
bool EditTextGet(string      &text,        // 텍스트
                 const long   chart_ID=0,  // 차트의 ID
                 const string name="Edit") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 텍스트 가져오기
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            ": 텍스트 가져오기 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 편집 개체 삭제                                                   |
//+------------------------------------------------------------------+
bool EditDelete(const long   chart_ID=0,  // 차트의 ID
                const string name="Edit") // 개체 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 레이블 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"편집₩ 개체 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 차트 창 크기
   long x_distance;
   long y_distance;
//--- 창 크기 설정
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("차트 너비 가져오기 실패! Error code = ",GetLastError());
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("차트 높이 가져오기 실패! Error code = ",GetLastError());
      return;
     }
//--- 편집 필드 변경 단계 지정
   int x_step=(int)x_distance/64;
//--- 편집 필드 좌표 및 해당 크기 설정
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- 텍스트를 로컬 변수에 저장
   string text=InpText;
//--- 편집 필드 생성
   if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText,InpFont,InpFontSize,InpAlign,I
      InpCorner,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,In
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 편집 필드를 늘이기
   while(x_size-x<x_distance*5/8)
     {
      //--- 편집 필드의 너비 증가
      x_size+=x_step;
      if(!EditChangeSize(0,InpName,x_size,y_size))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트를 다시 그리고 0.05 초 대기
      ChartRedraw();
      Sleep(50);
     }
//--- 0.5초 지연
   Sleep(500);
//--- 텍스트 변경
   for(int i=0;i<20;i++)
     {
      //--- 시작과 끝에 "+"를 추가
      text="+"+text+"+";
      if(!EditTextChange(0,InpName,text))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 차트를 다시 그리고 0.1초 대기
      ChartRedraw();
      Sleep(100);
     }
//--- 0.5초 지연
   Sleep(500);
//--- 편집 필드 삭제
   EditDelete(0,InpName);
   ChartRedraw();
//--- 1초 동안 대기
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_EVENT

이벤트 개체.

참조

마운스를 이벤트 위로 가져가면 해당 텍스트가 나타납니다.

예

다음 스크립트는 차트에서 이벤트 개체를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트가 ₩이벤트₩ 그래픽 개체를 그립니다."
#property description "고정점 날짜는 백분율로 설정됩니다."
#property description "차트 창 너비(막대 단위)."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string           InpName="Event";     // 이벤트 이름
input int              InpDate=25;          // 이벤트 날짜(%)
input string           InpText="Text";      // 이벤트 텍스트
input color            InpColor=clrRed;     // 이벤트 색상
input int              InpWidth=1;          // 강조 표시된 경우 점 크기
input bool             InpBack=false;       // 배경 이벤트
input bool             InpSelection=false; // 이동하려면 강조 표시
input bool             InpHidden=true;      // 개체 목록에 숨겨짐
```

© 2000-2025, MetaQuotes Ltd.

```
input long            InpZOrder=0;        // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 차트에 이벤트 개체 생성                                          |
//+------------------------------------------------------------------+
bool EventCreate(const long            chart_ID=0,       // 차트의 ID
                 const string          name="Event",     // 이벤트 이름
                 const int             sub_window=0,     // 하위 창 인덱스
                 const string          text="Text",      // 이벤트 텍스트
                 datetime              time=0,           // 시간
                 const color           clr=clrRed,       // 색상
                 const int             width=1,          // 강조 표시된 경우 점 너비
                 const bool            back=false,       // 배경에
                 const bool            selection=false, // 이동하려면 강조 표시
                 const bool            hidden=true,      // 개체 목록에 숨겨짐
                 const long            z_order=0)        // 마우스 클릭 우선 순위
  {
//--- 시간이 설정되지 않은 경우, 마지막 막대에 개체를 생성
   if(!time)
      time=TimeCurrent();
//--- 오류 값 재설정
   ResetLastError();
//--- 이벤트 개체 생성
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": \"이벤트₩ 개체 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 이벤트 텍스트 설정
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 개체가 강조 표시된 경우 고정점 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 이벤트 이동 모드를 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 개체 텍스트 변경                                          |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool EventTextChange(const long   chart_ID=0,   // 차트의 ID
                     const string name="Event", // 이벤트 이름
                     const string text="Text")  // 텍스트
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 텍스트 변경
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": 텍스트 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 개체 이동                                                 |
//+------------------------------------------------------------------+
bool EventMove(const long   chart_ID=0,   // 차트의 ID
               const string name="Event", // 이벤트 이름
               datetime     time=0)       // 시간
  {
//--- 시간이 설정되지 않은 경우 이벤트를 마지막 막대로 이동
   if(!time)
      time=TimeCurrent();
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 이동
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": \"이벤트₩ 개체 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 개체 삭제                                                 |
//+------------------------------------------------------------------+
bool EventDelete(const long   chart_ID=0,   // 차트의 ID
                 const string name="Event") // 이벤트 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 개체 삭제
   if(!ObjectDelete(chart_ID,name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print(__FUNCTION__,
            ": \"이벤트₩ 개체 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력 매개변수의 정확성 확인
   if(InpDate<0 || InpDate>100)
     {
      Print("오류! 입력 매개변수의 잘못된 값!");
      return;
     }
//--- 차트 창에 표시되는 막대 수
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- 사용할 날짜 값을 저장하기 위한 배열
//--- 이벤트 개체 고정점 좌표 설정 및 변경
   datetime date[];
//--- 메모리 할당
   ArrayResize(date,bars);
//--- 날짜 배열 채우기
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("시간 값 복사 실패! Error code = ",GetLastError());
      return;
     }
//--- 개체를 작성할 점 정의
   int d=InpDate*(bars-1)/100;
//--- 이벤트 개체 생성
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 이제 개체를 이동합니다
//--- 루프 카운터
   int h_steps=bars/2;
//--- 개체 이동
   for(int i=0;i<h_steps;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 다음의 값 사용
      if(d<bars-1)
         d+=1;
      //--- 점 이동
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트 다시 그리기
      ChartRedraw();
      // 0.05초 지연
      Sleep(50);
     }
//--- 1초 지연
   Sleep(1000);
//--- 차트에서 채널을 삭제
   EventDelete(0,InpName);
   ChartRedraw();
//--- 1초 지연
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

OBJ_RECTANGLE_LABEL

직사각형 레이블 개체.

참조

고정점 좌표는 픽셀 단위로 설정됩니다. 직사각형 레이블의 고정 모서리는 ENUM_BASE_CORNER 열거에서 선택할 수 있습니다. 직사각형 레이블의 테두리 유형은 ENUM_BORDER_TYPE 열거에서 선택할 수 있습니다.

개체는 사용자 지정 그래픽 인터페이스를 만들고 설계 하는 데 사용됩니다.

예

다음 스크립트는 차트에서 직사각형 레이블 개체를 만들고 이동합니다. 그래픽 개체의 속성을 만들고 변경할 수 있는 특수 기능이 개발되었습니다. 이러한 기능은 자체 애플리케이션에서 "있는 대로" 사용할 수 있습니다.

```
//--- 설명
#property description "스크립트는 ₩직사각형 레이블₩ 그래픽 개체를 생성합니다."
//--- 스크립트 실행 중 입력 매개변수의 표시 창
#property script_show_inputs
//--- 스크립트의 입력 매개변수
input string           InpName="RectLabel";          // 레이블 이름
input color            InpBackColor=clrSkyBlue;      // 배경 색상
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;        // 테두리 유형
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // 고정하기 위한 차트 모서리
input color            InpColor=clrDarkBlue;         // 평면 테두리 색상(평면)
input ENUM_LINE_STYLE  InpStyle=STYLE_SOLID;         // 평면 테두리 스타일(평면)
input int              InpLineWidth=3;               // 평면 테두리 너비(평면)
```

© 2000-2025, MetaQuotes Ltd.

```
input bool              InpBack=false;              // 배경 개체
input bool              InpSelection=true;          // 이동하려면 강조 표시
input bool              InpHidden=true;             // 개체 목록에 숨겨짐
input long              InpZOrder=0;                // 마우스 클릭 우선 순위
//+------------------------------------------------------------------+
//| 직사각형 레이블 생성 label                                        |
//+------------------------------------------------------------------+
bool RectLabelCreate(const long             chart_ID=0,               // 차트의 ID
                     const string           name="RectLabel",         // 레이블 이름
                     const int              sub_window=0,             // 하위 창 인덱스
                     const int              x=0,                      // X 좌표
                     const int              y=0,                      // Y 좌표
                     const int              width=50,                 // 너비
                     const int              height=18,                // 높이
                     const color            back_clr=C'236,233,216',  // 배경 색상
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,     // 테두리 유형
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // 고정을 위한 차
                     const color            clr=clrRed,               // 평면 테두리 색
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,        // 평면 테두리 스
                     const int              line_width=1,             // 평면 테두리 너
                     const bool             back=false,               // 배경에
                     const bool             selection=false,          // 이동하려면 강조
                     const bool             hidden=true,              // 개체 목록에 숨
                     const long             z_order=0)                // 마우스 클릭 우
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 직사각형 레이블 생성
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": 직사각형 레이블 생성 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 레이블 좌표 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- 레이블 크기 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- 배경 색상 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- 테두리 유형 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- 지정된 점 좌표를 기준으로 차트 모서리 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- 평면 테두리 색상 설정(평면 모드에서)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- 평면 테두리선 스타일 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- 평면 테두리 너비 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- 전경(false) 또는 배경(true)에 표시
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- 마우스를 사용하여 레이블 이동 모드 활성화(true) 또는 비활성화(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- 개체 목록에서 그래픽 개체 이름 숨기기(true) 또는 표시(false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- 차트에서 마우스 클릭 이벤트 수신 우선 순위 설정
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 직사각형 레이블 이동                                             |
//+------------------------------------------------------------------+
bool RectLabelMove(const long   chart_ID=0,       // 차트의 ID
                   const string name="RectLabel", // 레이블 이름
                   const int    x=0,              // X 좌표
                   const int    y=0)              // Y 좌표
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 직사각형 레이블 이동
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": 레이블의 X 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": 레이블의 Y 좌표 이동 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 직사각형 레이블 크기 변경                                        |
//+------------------------------------------------------------------+
bool RectLabelChangeSize(const long   chart_ID=0,       // 차트의 ID
                         const string name="RectLabel", // 레이블 이름
                         const int    width=50,         // 레이블 너비
                         const int    height=18)        // 레이블 높이
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 값 재설정
   ResetLastError();
//--- 레이블 크기 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": 레이블 너비 변경 실패! Error code = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": 레이블 높이 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 직사각형 레이블 테두리 유형 변경                                 |
//+------------------------------------------------------------------+
bool RectLabelChangeBorderType(const long             chart_ID=0,           // 차트의 I
                               const string           name="RectLabel",     // label n
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // 테두리 
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 테두리 유형 변경
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            ": 테두리 유형 변경 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 직사각형 레이블 삭제                                             |
//+------------------------------------------------------------------+
bool RectLabelDelete(const long   chart_ID=0,       // 차트의 ID
                     const string name="RectLabel") // 레이블 이름
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 레이블 삭제
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

© 2000-2025, MetaQuotes Ltd.

```
            ": 직사각형 레이블 삭제 실패! Error code = ",GetLastError());
      return(false);
     }
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 차트 창 크기
   long x_distance;
   long y_distance;
//--- 창 크기 설정
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("차트 너비 가져오기 실패! Error code = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("차트 높이 가져오기 실패! Error code = ",GetLastError());
      return;
     }
//--- 직사각형 레이블 좌표 지정
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- 레이블 크기 설정
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- 직사각형 레이블 생성
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- 차트를 다시 그리고 1초 대기
   ChartRedraw();
   Sleep(1000);
//--- 직사각형 레이블 크기 변경
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- 크기 조정
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 스크립트 작업이 강제로 비활성화 되었는지 확인
      if(IsStopped())
         return;
      //--- 차트를 다시 그리고 0.01 초 대기
      ChartRedraw();
      Sleep(10);
     }
//--- 1초 지연
   Sleep(1000);
//--- 테두리 유형 변경
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 테두리 유형 변경
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- 차트를 다시 그리고 1초 동안 대기
   ChartRedraw();
   Sleep(1000);
//--- 레이블 삭제
   RectLabelDelete(0,InpName);
   ChartRedraw();
//--- 1초 동안 대기
   Sleep(1000);
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

개체 속성

그래픽 개체는 개체 유형에 따라 다양한 속성을 가질 수 있습니다. 개체의 속성 값은 그래픽 개체 작업을 위한 함수(해당)에 의해 설정되고 수신됩니다.

기술적 분석에 사용되는 모든 개체는 추세선, 채널, 피보나치 도구 등 시간과 가격 좌표에 바인딩됩니다. 그러나 차트의 항상 표시되는 부분(기본 차트 창 또는 지표 하위 창)에 바인딩된 사용자 인터페이스를 개선하기 위한 여러 보조 개체가 있습니다.

개체	ID	X/Y	너비/높이	날짜/가격	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
텍스트	OBJ_TEXT	–	–	예	–	예	예
레이블	OBJ_LABEL	예	예(읽기 전용)	–	예	예	예
버튼	OBJ_BUTTON	예	예	–	예	–	–
비트맵	OBJ_BITMAP	–	예(읽기 전용)	예	–	예	–
비트맵 레이블	OBJ_BITMAP_LABEL	예	예(읽기 전용)	–	예	예	–
편집	OBJ_EDIT	예	예	–	예	–	–
직사각형 레이블	OBJ_RECTANGLE_LABEL	예	예	–	예	–	–

표에서 사용되는 명칭은 다음과 같습니다:

- **X/Y** – 차트 모서리에 상대적인 픽셀로 지정된 고정점 좌표;
- **너비/높이** – 개체의 너비와 높이를 가집니다. "읽기 전용"의 경우 개체가 차트에 렌더링된 후에만 너비와 높이 값이 계산됩니다;
- **날짜/가격** – 고정점 좌표는 날짜 및 가격 값을 사용하여 지정됩니다;
- **OBJPROP_CORNER** – 고정점 좌표가 지정되는 차트 모서리를 정의합니다. ENUM_BASE_CORNER 열거의 4개 값 중 하나일 수 있습니다;
- **OBJPROP_ANCHOR** – 개체 자체에 고정점을 정의하며, ENUM_ANCHOR_POINT 열거의 9개 값 중 하나일 수 있습니다. 좌표는 바로 이 지점에 선택한 차트 모서리까지 지정됩니다;
- **OBJPROP_ANGLE** – 개체 회전 각도를 시계 반대 방향으로 지정합니다.

차트를 따라 개체를 만들고 이동하기 위한 ObjectCreate() 및 ObjectMove() 작업과 그래픽 개체의 속성을 정의하는 기능은 실제로 차트에 명령을 보내는 데 사용됩니다. 이러한 기능이 성공적으로 실행되면 명령이 차트 이벤트의 공통 대기열에 포함됩니다. 차트의 이벤트 대기열을 처리할 때 그래픽 개체의 특성에 대한 시각적 변화가 구현됩니다.

따라서 이러한 기능을 호출한 후 그래픽 개체의 즉각적인 시각적 업데이트를 기대하지 마십시오. 일반적으로 차트의 그래픽 개체는 새 시세 도착, 차트 창 크기 조정 등 변경 이벤트 이후에 터미널에 의해 자동으로 업데이트됩니다. ChartRedraw() 기능을 사용하여 그래픽 개체를 강제로 업데이트합니다.

© 2000-2025, MetaQuotes Ltd.

ObjectSetInteger() 및 ObjectGetInteger() 함수의 경우

ENUM_OBJECT_PROPERTY_INTEGER

식별자	설명	속성 유형
OBJPROP_COLOR	색상	색상
OBJPROP_STYLE	스타일	ENUM_LINE_STYLE
OBJPROP_WIDTH	선 굵기	int
OBJPROP_BACK	배경에 개체	bool
OBJPROP_ZORDER	차트를 클릭하는 이벤트를 수신하기 위한 그래픽 개체의 우선순위 (CHARTEVENT_CLICK). 개체를 만들 때 기본 0 값이 설정되며, 필요한 경우 우선순위	long

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	를 높일 수 있습니다. 개체를 서로 맨 위에 배치하면 우선순위가 가장 높은 개체 중 하나만 CHARTEVENT_CLICK 이벤트를 받게 됩니다.	
OBJPROP_FILL	개체 색상 채우기 (for OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPS	bool

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	E, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	
OBJPROP_HIDDEN	터미널 메뉴 "차트" - "개체" - "개체 목록" 에서 개체목록에 트래픽 개체 이름을 표시하는 것을 금지합니다. True 값을 사용하면	bool

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	목록에서 개체를 숨길 수 있습니다. 기본적으로 true는 일정 이벤트, 거래 기록을 표시하는 개체 및 MQL5 프로그램에서 생성된 개체로 설정됩니다. 이러한 그래픽 개체를 보고 해당 속성에 액세스하려면 "개	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	체목록" 창에서 "전체" 버튼을 클릭하십시오.	
OBJPROP_SELECTED	개체가 선택되었습니다	bool
OBJPROP_READONLY	편집 개체에서 텍스트를 편집할 수 있는 기능	bool
OBJPROP_TYPE	개체 유형	ENUM_OBJECT r/o
OBJPROP_TIME	시간 좌표	datetime 수정자=고정점 수
OBJPROP_SELECTABLE	개체 가용성	bool
OBJPROP_CREATETIME	개체 생성 시간	datetime r/o
OBJPROP_LEVELS	레벨 수	int
OBJPROP_LEVELCOLOR	선 레벨 색상	색상 수정자=레벨 숫자

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
OBJPROP_LEVELSTYLE	선 레벨 스타일	ENUM_LINE_STYLE 수정자=레벨 숫자
OBJPROP_LEVELWIDTH	선 레벨 굵기	int 수정자=레벨 수
OBJPROP_ALIGN	"편집" 개체의 수평 텍스트 정렬 (OBJ_EDIT)	ENUM_ALIGN_MODE
OBJPROP_FONTSIZE	글꼴 크기	int
OBJPROP_RAY_LEFT	선 좌측으로 향함	bool
OBJPROP_RAY_RIGHT	선 우측으로 향함	bool
OBJPROP_RAY	수직선이 차트의 모든 창을 통과합니다	bool
OBJPROP_ELLIPSE	피보나치 아크 개체의 전체 타원 표시	bool

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	(OBJ_FIBOARC)	
OBJPROP_ARROWCODE	화살표 개체의 화살표 코드	uchar
OBJPROP_TIMEFRAMES	시간 프레임에 개체 가시성	플래그의 세트 flags
OBJPROP_ANCHOR	그래픽 객체의 고정점 위치	ENUM_ARROW_ANCHOR (OBJ_ARROW 경우), ENUM_ANCHOR_POINT (OBJ_LABEL, OBJ_BITMAP_LABEL 및 OBJ_TEXT 경우)
OBJPROP_XDISTANCE	바인딩 모서리 X 축을 따라 픽셀 단위로 표시되는 거리 (참고 참조)	int
OBJPROP_YDISTANCE	바인딩 모서리에서 Y 측을	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	따라 픽셀 단위로 표시되는 거리 (참고 참조)	
OBJPROP_DIRECTION	갠 개체 추세	ENUM_GANN_DIRECTION
OBJPROP_DEGREE	엘리엇 파동 표시 레벨	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	엘리엇 웨이브 표시선 표시	bool
OBJPROP_STATE	버튼 상태 (누름/해제)	bool
OBJPROP_CHART_ID	"차트" 개체의 ID (OBJ_CHART). 차트 작업에 설명된	long r/o

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	함수를 사용하여 일반 차트와 같이 이 개체의 속성을 사용할 수 있지만 일부 예외가 있습니다.	
OBJPROP_XSIZE	X 축을 따라 있는 개체의 너비(픽셀). OBJ_LABEL (읽기 전용), OBJ_BUTTON, OBJ_CHART, OBJ	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL 개체에 대해 지정됨.	
OBJPROP_YSIZE	Y 축을 따라 있는 개체의 높이(픽셀) OBJ_LABEL(읽기 전용), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL 개체에 대해 지정됨.	
OBJPROP_XOFFSET	그래픽 개체 "비트맵 레이블" 및 "비트맵"에서 직사각형 표시 영역 좌측 상단 모서리의 X 좌표 (OBJ_BITMAP_LABEL 및 OBJ	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	_BITMAP). 이 값은 원본 이미지의 좌측 상단 모서리에 상대적인 픽셀로 설정됩니다.	
OBJPROP_YOFFSET	그래픽 개체 "비트맵 레이블" 및 "비트맵" (OBJ_BITMAP_LABEL 및 OBJ_BITMAP)에서 직사각형 표시 영역 좌측 상단 모서리에	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	있는 Y 좌표. 이 값은 원본 이미지의 좌측 상단 모서리에 상대적인 픽셀로 설정됩니다.	
OBJPROP_PERIOD	차트 개체의 시간 프레임	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	차트 개체의 시간 척도 표시	bool
OBJPROP_PRICE_SCALE	차트 개체의 가격 스케일 표시	bool
OBJPROP_CHART_SCALE	차트 개체의 스케일	int 0~5 범위의 값
OBJPROP_BGCOLOR	OBJ_EDIT,	색상

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	OBJ_BUTTON, OBJ_RECTANGLE_LABEL의 배경색	
OBJPROP_CORNER	그래픽 개체를 연결하기 위한 차트 모서리	ENUM_BASE_CORNER
OBJPROP_BORDER_TYPE	"직사각형 레이블" 개체의 테두리 유형	ENUM_BORDER_TYPE
OBJPROP_BORDER_COLOR	OBJ_EDIT 및 OBJ_BUTTON 개체의 테두리 색상	색상

차트 작업을 "차트" 개체 (OBJ_CHART)에 대해 사용할 경우 다음과 같은 제한이 적용됩니다:

- ChartClose()를 사용하여 닫을 수 없습니다;

© 2000-2025, MetaQuotes Ltd.

- 기호/주기는 ChartSetSymbolPeriod() 함수를 사용하여 변경할 수 없습니다;
- 다음 속성은 효과적이지 않습니다 CHART_SCALE, CHART_BRING_TO_TOP, CHART_SHOW_DATE_SCALE 및 CHART_SHOW_PRICE_SCALE (ENUM_CHART_PROPERTY_INTEGER).

OBJ_BITMAP_LABEL 및 OBJ_BITMAP 개체에 대해 이미지 표시 특수 모드를 설정할 수 있습니다. 이 모드에서는 원본 이미지의 일부(직사각형 가시 영역이 적용된 부분)만 표시되고 나머지 이미지는 보이지 않게 됩니다. 이 영역의 크기는 OBJPROP_XSIZE 및 OBJPROP_YSIZE 속성을 사용하여 설정해야 합니다. 표시되는 영역은 OBJPROP_XOFFSET 및 OBJPROP_YOFFSET 속성을 사용하여 원래 이미지 내에서만 "이동"할 수 있습니다.

고정된 크기의 개체 OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT 및 OBJ_CHART의 경우, OBJPROP_XDISTANCE 및 OBJPROP_YDISTANCE 속성은 차트 모서리(OBJPROP_CORNER)를 기준으로 개체의 좌측 상단 위치를 설정합니다.

다음 함수의 경우: ObjectSetDouble() 및 ObjectGetDouble()

ENUM_OBJECT_PROPERTY_DOUBLE

식별자	설명	속성 유형
OBJPROP_PRICE	가격 좌표	double 수정자=고정점 수
OBJPROP_LEVELVALUE	레벨 값	double 수정자=레벨 수
OBJPROP_SCALE	스케일 (갠 개체 및 피보나치 아크 특성)	double
OBJPROP_ANGLE	각도. 프로그램에서 생성된 각도가 지정되지	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	않은 개체의 경우 값은 EMPTY_VALUE 와 같습니다.	
OBJPROP_DEVIATION	표준 편차 채널에 대한 편차	double

다음 함수의 경우: ObjectSetString() 및 ObjectGetString()

ENUM_OBJECT_PROPERTY_STRING

식별자	설명	속성 유형
OBJPROP_NAME	개체 이름	string
OBJPROP_TEXT	개체에 대한 설명 (개체에 포함된 텍스트)	string
OBJPROP_TOOLTIP	도구 설명의 텍스트. 속성이 설정	string

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	되지 않은 경우 터미널에서 자동으로 생성된 도구 설명이 표시됩니다. 도구 설명에 "\n" (라인 피드) 값을 설정하여 도구 설명을 비활성화할 수 있습니다.	
OBJPROP_LEVELTEXT	레벨 설명	string 수정자=레벨 수
OBJPROP_FONT	글꼴	string
OBJPROP_BMPFILE	비트맵 레이블의 BMP 파일 이름	string modifier: 0-state ON, 1-state OFF

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 유형
	입니다. 추가 참조 Resources	
OBJPROP_SYMBOL	차트 개체 기호	string

OBJ_RECTANGLE_LABEL 개체의 경우("직사각형 레이블") 세 가지 설계 모드 중 하나를 설정할 수 있으며, 여기에 해당하는 ENUM_BORDER_TYPE 값을 지정할 수 있습니다.

ENUM_BORDER_TYPE

식별자	설명
BORDER_FLAT	평면 형태
BORDER_RAISED	볼록한 형태
BORDER_SUNKEN	오목한 형태

OBJ_EDIT 개체("편집") 및 ChartScreenShot() 함수의 경우 ENUM_ALIGN_MODE 열거 값을 사용하여 수평 정렬 유형을 지정할 수 있습니다.

ENUM_ALIGN_MODE

식별자	설명
ALIGN_LEFT	좌측 정렬
ALIGN_CENTER	중앙 (편집 개체 전용)
ALIGN_RIGHT	우측 정렬

예:

```
#define  UP          "\x0431"
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
    {
     Print("Object ",label_name," not found. Error code = ",GetLastError());
     //--- 레이블 개체 생성
     ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
     //--- X 좌표 설정
     ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
     //--- Y 좌표 설정
     ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
     //--- 텍스트 색상 지정
     ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
     //--- 개체 레이블 텍스트 정의
     ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
     //--- 글꼴 지정
     ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
     //--- 글꼴 크기 지정
     ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
     //--- 시계방향 45도 회전
     ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
     //--- 마우스 선택 불가능
     ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
     //--- 차트에 그리기
     ChartRedraw(0);
    }
}
```

© 2000-2025, MetaQuotes Ltd.

개체 바인딩 방법

Graphical objects Text, Label, Bitmap and Bitmap Label 그래픽 개체 텍스트, 레이블, 비트맵 및 비트맵 레이블 (OBJ_TEXT, OBJ_LABEL, OBJ_BITMAP 및 OBJ_BITMAP_LABEL)은 OBJPROP_ANCHOR 속성에 의해 정의된 9가지 좌표 바인딩 방법 중 하나를 가질 수 있습니다.

개체	ID	X/Y	너비/높이	날짜/가격	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
텍스트	OBJ_TEXT	–	–	예	–	예	예
레이블	OBJ_LABEL	예	예(읽기 전용)	–	예	예	예
버튼	OBJ_BUTTON	예	예	–	예	–	–
비트맵	OBJ_BITMAP	–	예(읽기 전용)	예	–	예	–
비트맵 레이블	OBJ_BITMAP_LABEL	예	예(읽기 전용)	–	예	예	–
편집	OBJ_EDIT	예	예	–	예	–	–
직사각형 레이블	OBJ_RECTANGLE_LABEL	예	예	–	예	–	–

표에서 사용되는 명칭은 다음과 같습니다:

- **X/Y** – 차트 모서리에 상대적인 픽셀로 지정된 고정점 좌표;
- **너비/높이** – 개체의 너비와 높이를 가집니다. "읽기 전용"의 경우 개체가 차트에 렌더링된 후에만 너비와 높이 값이 계산됩니다;
- **날짜/가격** – 고정점 좌표는 날짜 및 가격 값을 사용하여 지정됩니다;
- **OBJPROP_CORNER** – 고정점 좌표가 지정되는 차트 모서리를 정의합니다. ENUM_BASE_CORNER 열거의 4개 값 중 하나일 수 있습니다;
- **OBJPROP_ANCHOR** – 개체 자체에 고정점을 정의하며, ENUM_ANCHOR_POINT 열거의 9개 값 중 하나일 수 있습니다. 좌표는 바로 이 지점에 선택한 차트 모서리까지 지정됩니다;
- **OBJPROP_ANGLE** – 개체 회전 각도를 시계 반대 방향으로 지정합니다.

필요한 번영은 ObjectSetInteger 함수(chart_handle, object_name, OBJPROP_ANCHOR, anchor_point_mode)를 사용하여 지정할 수 있습니다. 여기서 anchor_point_mode는 ENUM_ANCHOR_POINT 값 중 하나입니다.

ENUM_ANCHOR_POINT

ID	설명
ANCHOR_LEFT_UPPER	좌측 상단 모서리의 고정점

© 2000-2025, MetaQuotes Ltd.

ID	설명
ANCHOR_LEFT	가운데의 좌측에 고정점
ANCHOR_LEFT_LOWER	좌측 하단 모서리의 고정점
ANCHOR_LOWER	가운데 아래의 고정점
ANCHOR_RIGHT_LOWER	우측 하단 모서리의 고정점
ANCHOR_RIGHT	가운데 우측의 고정점
ANCHOR_RIGHT_UPPER	우측 상단 모서리의 고정점
ANCHOR_UPPER	가운데 위의 고정점
ANCHOR_CENTER	고정점이 개체의 가운데에 정확히 위치

OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT 및 OBJ_CHART 개체의 좌측 상단 모서리(ANCHOR_LEFT_UPPER)에 고정점이 고정되어 있습니다.

예:

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
  {
   Print("Object ",text_name," not found. Error code = ",GetLastError());
   //--- 차트의 최대값 가져오기
   double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
   //--- 개체 레이블 생성
   ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
   //--- 글씨 색상 설정
   ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
   //--- 배경 색상 설정
   ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
   //--- 레이블 개체의 텍스트 설정
   ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
   //--- 글씨 글꼴 설정
   ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
   //--- 글꼴 크기 설정
   ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
   //--- 상단 우측 모서리에 바인딩
   ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
   //--- 시계 반대 방향으로 90도 회전
   ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
   //--- 마우스를 통한 개체 선택 금지
   ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
   //--- 개체 다시 그리기
   ChartRedraw(0);
  }
```

그래픽 개체 화살표(OBJ_ARROW)에는 좌표를 연결하는 두 가지 방법만 있습니다. 식별자는 ENUM_ARROW_ANCHOR에 나열됩니다.

© 2000-2025, MetaQuotes Ltd.

ENUM_ARROW_ANCHOR

ID	설명
ANCHOR_TOP	상부 측면 앵커
ANCHOR_BOTTOM	하부 측면 앵커

예:

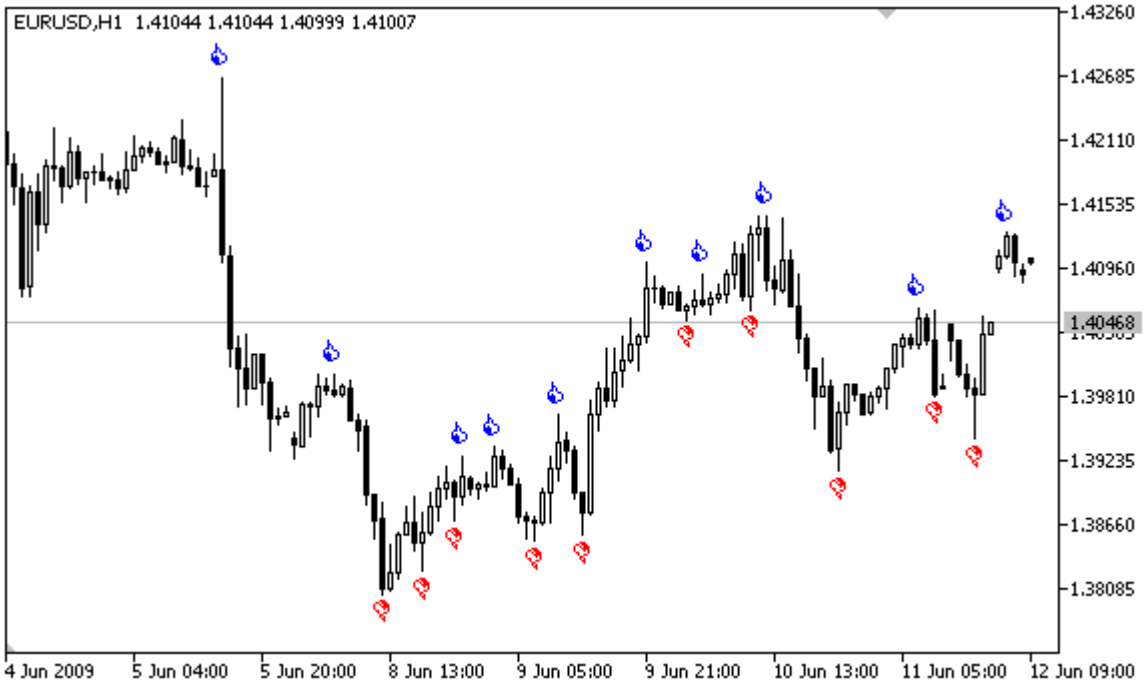
```
void OnStart()
  {
//--- 보조 배열
   double Ups[],Downs[];
   datetime Time[];
//--- 배열을 시계열로 설정
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- 지표 프랙탈 핸들 생성
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle = ",FractalsHandle);
//--- 마지막 오류 값을 0으로 설정
   ResetLastError();
//--- 지표 값 복사 시도
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("상위 프랙탈을 복사할 수 없습니다. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- 지표 값 복사 시도
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("하위 프랙탈을 복사할 수 없습니다. Error = ",GetLastError());
      return;
     }

   ResetLastError();
//--- 마지막 1000개의 시작 막대가 포함된 시계열 복사
   copied=CopyTime(NULL,0,0,1000,Time);
   if(copied<=0)
     {
      Print("마지막 100개 막대 시작 시간을 복사할 수 없습니다.");
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   int upcounter=0,downcounter=0; // 화살표 수를 파악
   bool created;// 개체 생성 시도의 결과를 수신
   for(int i=2;i<copied;i++)// 지표 iFractals 값 실행
     {
      if(Ups[i]!=EMPTY_VALUE)// 상위 프랙탈 발견
        {
         if(upcounter<10)// 10개 이하의 "위쪽" 화살표 생성
           {
            //--- "위쪽" 개체 생성 시도
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
            if(created)// 설정된 경우 - 튜닝을 수행합니다
              {
               //--- 막대를 덮지 않기 위해 점 앵커가 아래에 있습니다
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
               //--- 최종 터치 - 색칠
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
               upcounter++;
              }
           }
        }
      if(Downs[i]!=EMPTY_VALUE)// 하부 프랙탈 발견
        {
         if(downcounter<10)// 10개 이하의 "아래로" 화살표 만들기
           {
            //--- "아래쪽" 개체 생성 시도
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Down
            if(created)// 설정된 경우 - 튜닝을 수행합니다
              {
               //--- 막대를 덮지 않기 위해 점 앵커가 위에 있습니다
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
               //--- 최종 터치 - 색칠
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
               downcounter++;
              }
           }
        }
     }
  }
```

스크립트 실행 후 차트는 이 그림과 같이 표시됩니다.

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

개체가 첨부된 차트 모서리

차트 모서리를 설정할 수 있는 여러 그래픽 개체가 있으며, 이 개체에 상대적인 좌표는 픽셀 단위로 지정됩니다. 다음은 개체의 유형입니다 (괄호 안에 개체 유형 식별자가 지정됨):

- 레이블 (OBJ_LABEL);
- 버튼 (OBJ_BUTTON);
- 비트맵 레이블 (OBJ_BITMAP_LABEL);
- 편집 (OBJ_EDIT).
- 직사각형 레이블 (OBJ_RECTANGLE_LABEL);

개체	ID	X/Y	너비/높이	날짜/가격	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
텍스트	OBJ_TEXT	–	–	예	–	예	예
레이블	OBJ_LABEL	예	예(읽기 전용)	–	예	예	예
버튼	OBJ_BUTTON	예	예	–	예	–	–
비트맵	OBJ_BITMAP	–	예(읽기 전용)	예	–	예	–
비트맵 레이블	OBJ_BITMAP_LABEL	예	예(읽기 전용)	–	예	예	–
편집	OBJ_EDIT	예	예	–	예	–	–
직사각형 레이블	OBJ_RECTANGLE_LABEL	예	예	–	예	–	–

표에서 사용되는 명칭은 다음과 같습니다:

- **X/Y** – 차트 모서리에 상대적인 픽셀로 지정된 고정점 좌표;
- **너비/높이** – 개체의 너비와 높이를 가집니다. "읽기 전용"의 경우 개체가 차트에 렌더링된 후에만 너비와 높이 값이 계산됩니다;
- **날짜/가격** – 고정점 좌표는 날짜 및 가격 값을 사용하여 지정됩니다;
- **OBJPROP_CORNER** – 고정점 좌표가 지정되는 차트 모서리를 정의합니다. ENUM_BASE_CORNER 열거의 4개 값 중 하나일 수 있습니다;
- **OBJPROP_ANCHOR** – 개체 자체에 고정점을 정의하며, ENUM_ANCHOR_POINT 열거의 9개 값 중 하나일 수 있습니다. 좌표는 바로 이 지점에 선택한 차트 모서리까지 지정됩니다;
- **OBJPROP_ANGLE** – 개체 회전 각도를 시계 반대 방향으로 지정합니다.

X 및 Y 좌표를 픽셀 단위로 측정할 차트 모서리를 지정하려면 다음에서 ObjectSetInteger(chartID, name, OBJPROP_CORNER, **chart_corner**)를 사용합니다:

- chartID – 차트 식별자;

© 2000-2025, MetaQuotes Ltd.

- name – 그래픽 개체의 이름;
- OBJPROP_CORNER – 바인딩 모서리를 지정하는 속성 ID;
- **chart_corner** – 원하는 차트 코너는 ENUM_BASE_CORNER 열거값 중 하나일 수 있습니다.

ENUM_BASE_CORNER

ID	설명
CORNER_LEFT_UPPER	좌표 중심은 차트의 좌측 상단 모서리에 있습니다
CORNER_LEFT_LOWER	좌표 중심은 차트의 좌측 하단 모서리에 있습니다
CORNER_RIGHT_LOWER	좌표 중심은 차트의 우측 하단 모서리에 있습니다
CORNER_RIGHT_UPPE	좌표 중심은 차트의 우측 상단 모서리에 있습니다

예:

```
void CreateLabel(long   chart_id,
                 string name,
                 int    chart_corner,
                 int    anchor_point,
                 string text_label,
                 int    x_ord,
                 int    y_ord)
  {
//---
   if(ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0))
     {
      ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner);
      ObjectSetInteger(chart_id,name,OBJPROP_ANCHOR,anchor_point);
      ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
      ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
      ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
     }
   else
      Print("개체를 생성하는데 실패 OBJ_LABEL ",name,", Error code = ", GetLastError());
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int height=(int)ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=(int)ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
   CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,ANCHOR_LEFT_UPPER,arrows[0],50,50);
   CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,ANCHOR_RIGHT_UPPER,arrows[1],50,50);
   CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,ANCHOR_RIGHT_LOWER,arrows[2],50,50);
   CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,ANCHOR_LEFT_LOWER,arrows[3],50,50);
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

개체 가시성

개체 가시성 플래그의 조합에 따라 개체가 표시되는 차트 시간프레임이 결정됩니다. OBJPROP_TIMEFRAMES 속성 값을 설정/가져오기 위해 ObjectSetInteger()/ObjectGetInteger() 함수를 사용할 수 있습니다.

ID	값	설명
OBJ_NO_PERIODS	0	개체가 일부 시간프레임에 그려지지 않습니다
OBJ_PERIOD_M1	0x00000001	개체가 1분 차트에 그려집니다
OBJ_PERIOD_M2	0x00000002	개체가 2분 차트에 그려집니다
OBJ_PERIOD_M3	0x00000004	개체가 3분 차트에 그려집니다
OBJ_PERIOD_M4	0x00000008	개체가 4분 차트에 그려집니다
OBJ_PERIOD_M5	0x00000010	개체가 5분 차트에 그려집니다
OBJ_PERIOD_M6	0x00000020	개체가 6분 차트에 그려집니다
OBJ_PERIOD_M10	0x00000040	개체가 10분 차트에 그려집니다
OBJ_PERIOD_M12	0x00000080	개체가 12분 차트에 그려집니다
OBJ_PERIOD_M15	0x00000100	개체가 15분 차트에 그려집니다
OBJ_PERIOD_M20	0x00000200	개체가 20분 차트에 그려집니다
OBJ_PERIOD_M30	0x00000400	개체가 30분 차트에 그려집니다
OBJ_PERIOD_H1	0x00000800	개체가 1시간 차트에 그려집니다
OBJ_PERIOD_H2	0x00001000	개체가 2시간 차트에 그려집니다
OBJ_PERIOD_H3	0x00002000	개체가 3시간 차트에 그려집니다
OBJ_PERIOD_H4	0x00004000	개체가 4시간 차트에 그려집니다
OBJ_PERIOD_H6	0x00008000	개체가 6시간 차트에 그려집니다,
OBJ_PERIOD_H8	0x00010000	개체가 8시간 차트에 그려집니다
OBJ_PERIOD_H12	0x00020000	개체가 12시간 차트에 그려집니다
OBJ_PERIOD_D1	0x00040000	개체가 일 차트에 그려집니다
OBJ_PERIOD_W1	0x00080000	개체가 주 차트에 그려집니다
OBJ_PERIOD_MN1	0x00100000	개체가 월 차트에 그려집니다

© 2000-2025, MetaQuotes Ltd.

ID	값	설명
OBJ_ALL_PERIODS	0x001fffff	개체가 모든 시간대에 그려집니다

가시성 플래그는 "|" 기호를 사용하여 조합할 수 있으며, 예를 들어 OBJ_PERIOD_M10|OBJ_PERIOD_H4 플래그의 조합은 개체를 10분 및 4시간 시간 프레임에서 볼 수 있음을 의미합니다.

예:

```
void OnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;           // 전날 고가
   double prevLow;            // 전날 저가
   double highs[],lows[];     // 고가 및 저가 배열

//--- 마지막 오류 재설정
   ResetLastError();
//--- 일별 기간에서 마지막 2개의 높은 값을 가져오기
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // 복사가 성공인 경우
     {
      Print("지난 이틀간 고가를 성공적으로 가져왔습니다");
      prevHigh=highs[0]; // 전날 고가
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // 상위레벨 이름의 개체를 찾을 수 없습니다
        {
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // 수평선 개체 생성
        }
      //--- 선 상위레벨의 가격 레벨 값 설정
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
      //--- 가시성을 PERIOD_M10 및 PERIOD_H4로만 설정
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
     {
      Print("지난 이틀간 고가를 가져올 수 없습니다, 오류 = ",GetLastError());
     }

//--- 마지막 오류 재설정
   ResetLastError();
//--- 일별 기간에서 이틀간 저가 값 가져오기
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // 복사가 성공인 경우
     {
      Print("지난 이틀간 저가를 성공적으로 가져왔습니다");
      prevLow=lows[0]; // 전날 저가
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("prevLow = ",prevLow);
      if(ObjectFind(0,lowlevel)<0) // 하위레벨 이름의 개체를 찾을 수 없습니다
        {
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // 수평선 개체 생성
        }
      //--- 하위레벨 선에 대한 가격 레벨 값 설정
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- 가시성을 PERIOD_M10 및 PERIOD_H4로만 설정
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else Print("지난 이틀간 저가를 가져올 수 없습니다, 오류 = ",GetLastError());

   ChartRedraw(0); // 차트 강제 다시 그리기
  }
```

추가 참조

PeriodSeconds, Period, Chart timeframes, Date and Time

© 2000-2025, MetaQuotes Ltd.

엘리엇 파동 레벨

엘리엇 파동은 OBJ_ELLIOTWAVE5 및 OBJ_ELLIOTWAVE3 유형의 두 개의 그래픽 개체로 표시됩니다. 파동의 크기(파동 라벨링 방법)를 설정하려면 ENUM_ELLIOT_WAVE_DEGREE 열거값 중 하나를 할당할 수 있는 OBJPROP_DEGREE 속성이 사용됩니다.

ENUM_ELLIOT_WAVE_DEGREE

ID	설명
ELLIOTT_GRAND_SUPERCYCLE	그랜드 슈퍼사이클
ELLIOTT_SUPERCYCLE	슈퍼사이클
ELLIOTT_CYCLE	사이클
ELLIOTT_PRIMARY	초급
ELLIOTT_INTERMEDIATE	중급
ELLIOTT_MINOR	마이너
ELLIOTT_MINUTE	미닛
ELLIOTT_MINUETTE	미뉴엣
ELLIOTT_SUBMINUETTE	서브미뉴엣

예:

```
   for(int i=0;i<ObjectsTotal(0);i++)
     {
      string currobj=ObjectName(0,i);
      if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3) ||
          ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
        {
         //--- INTERMEDIATE에서 표시 수준을 설정
         ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
         //--- 파동 사이에 선을 표시
         ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
         //--- 선 색상 설정
         ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
         //--- 선 굵기 설정
         ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
         //--- 설명 설정
         ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

갠(Gann) 개체

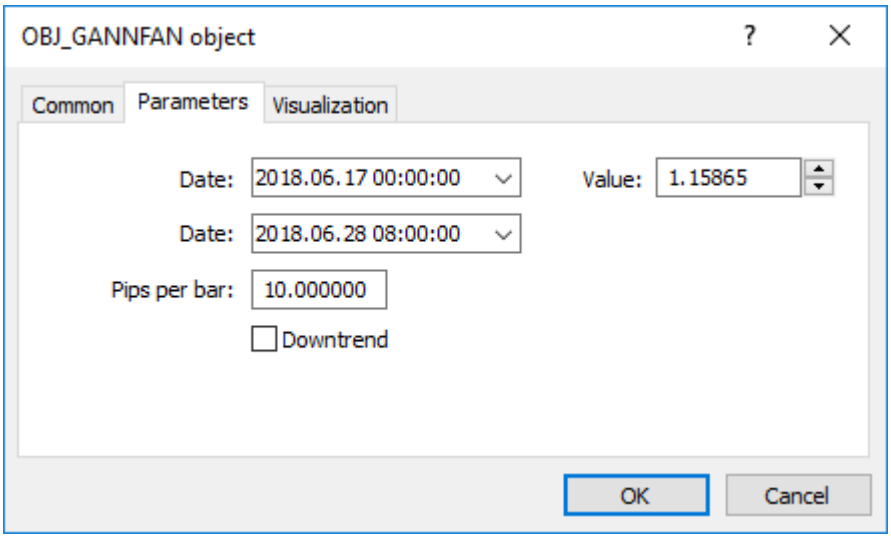
갠 팬 (OBJ_GANNFAN) 및 갠 그리드 (OBJ_GANNGRID) 개체의 경우 추세 방향을 설정하는 ENUM_GANN_DIRECTION 열거의 두 값을 지정할 수 있습니다.

ENUM_GANN_DIRECTION

ID	설명
GANN_UP_TREND	상향 추세선에 해당하는 선
GANN_DOWN_TREND	하향 추세선에 해당하는 선

주선 스케일을 1x1로 설정하려면, ObjectSetDouble(chart_handle, gann_object_name, OBJPROP_SCALE, scale) 함수를 사용합니다:

- chart_handle – 개체가 위치한 차트 창;
- gann_object_name – 개체 이름;
- OBJPROP_SCALE – "스케일" 속성의 식별자;
- scale – 필요한 스케일(Pips/Bar 단위).



갠 팬 생성 예제:

```
void OnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// 개체를 찾을 수 없습니다
     {
      //--- 오류에 대한 정보
      Print("개체 ",my_gann," not found. Error code = ",GetLastError());
      //--- 차트의 최대값 가져오기
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- 차트의 최소 가격을 가져오기
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
      //--- 차트에 몇 개의 막대가 표시됩니까?
```

© 2000-2025, MetaQuotes Ltd.

```
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- 배열을 작성하고 각 막대의 시작 시간을 기록합니다
      datetime Time[];
      //--- 배열에 대한 액세스를 시계열의 액세스로 정렬
      ArraySetAsSeries(Time,true);
      //--- 이제 차트에 표시되는 막대의 데이터를 배열로 복사합니다
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("시작 시간으로 배열을 복사할 수 없습니다!");
         return;
        }
      //--- 예비준비 완료

      //--- 차트의 중앙 막대 인덱스
      int center_bar=bars_on_chart/2;
      //--- 차트 적도 - 최대값과 최소값 사이
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- 첫 번째 고정점의 좌표를 중심으로 설정
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- 우측에 있는 두 번째 고정점
                   Time[center_bar/2],(mean+chart_min_price)/2.0);
      Print("Time[center_bar] = "+(string)Time[center_bar]+"  Time[center_bar/2] = "+
      //Print("Time[center_bar]/="+Time[center_bar]+"  Time[center_bar/2]="+Time[cente
      //--- 스캐일을 Pips / Bar 단위로 설정
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- 선 추세 설정
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- 선 굵기 설정
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- 선 스타일 정의
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- 선 색상 설정
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- 사용자가 개체를 선택할 수 있도록 허용
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- 직접 선택하세요
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- 차트에 그리기
      ChartRedraw(0);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

웹 색상

다음에 대해 아래와 같은 색상 상수가 정의됩니다 –색상 유형:

clrBlack	clrDarkGreen	clrDarkSlateGray	clrOlive	clrGreen	clrTeal	clrNavy	clrPurple
clrMaroon	clrIndigo	clrMidnightBlue	clrDarkBlue	clrDarkOliveGreen	clrSaddleBrown	clrForestGreen	clrOliveDrab
clrSeaGreen	clrDarkGoldenrod	clrDarkSlateBlue	clrSienna	clrMediumBlue	clrBrown	clrDarkTurquoise	clrDimGray
clrLightSeaGreen	clrDarkViolet	clrFireBrick	clrMediumVioletRed	clrMediumSeaGreen	clrChocolate	clrCrimson	clrSteelBlue
clrGoldenrod	clrMediumSpringGreen	clrLawnGreen	clrCadetBlue	clrDarkOrchid	clrYellowGreen	clrLimeGreen	clrOrangeRed
clrDarkOrange	clrOrange	clrGold	clrYellow	clrChartreuse	clrLime	clrSpringGreen	clrAqua
clrDeepSkyBlue	clrBlue	clrMagenta	clrRed	clrGray	clrSlateGray	clrPeru	clrBlueViolet
clrLightSlateGray	clrDeepPink	clrMediumTurquoise	clrDodgerBlue	clrTurquoise	clrRoyalBlue	clrSlateBlue	clrDarkKhaki
clrIndianRed	clrMediumOrchid	clrGreenYellow	clrMediumAquamarine	clrDarkSeaGreen	clrTomato	clrRosyBrown	clrOrchid
clrMediumPurple	clrPaleVioletRed	clrCoral	clrCornflowerBlue	clrDarkGray	clrSandyBrown	clrMediumSlateBlue	clrTan
clrDarkSalmon	clrBurlyWood	clrHotPink	clrSalmon	clrViolet	clrLightCoral	clrSkyBlue	clrLightSalmon
clrPlum	clrKhaki	clrLightGreen	clrAquamarine	clrSilver	clrLightSkyBlue	clrLightSteelBlue	clrLightBlue
clrPaleGreen	clrThistle	clrPowderBlue	clrPaleGoldenrod	clrPaleTurquoise	clrLightGray	clrWheat	clrNavajoWhite
clrMoccasin	clrLightPink	clrGainsboro	clrPeachPuff	clrPink	clrBisque	clrLightGoldenrod	clrBlanchedAlmond
clrLemonChiffon	clrBeige	clrAntiqueWhite	clrPapayaWhip	clrCornsilk	clrLightYellow	clrLightCyan	clrLinen
clrLavender	clrMistyRose	clrOldLace	clrWhiteSmoke	clrSeashell	clrIvory	clrHoneydew	clrAliceBlue
clrLavenderBlush	clrMintCream	clrSnow	clrWhite				

ObjectSetInteger() 함수를 사용하여 색상을 개체로 설정할 수 있습니다. 사용자 지정 지표로 색상을 설정하려면 PlotIndexSetInteger() 함수를 사용합니다. 색상 값을 가져오는 데 유사한 함수 ObjectGetInteger() 및 PlotIndexGetInteger()이 있습니다.

© 2000–2025, MetaQuotes Ltd.

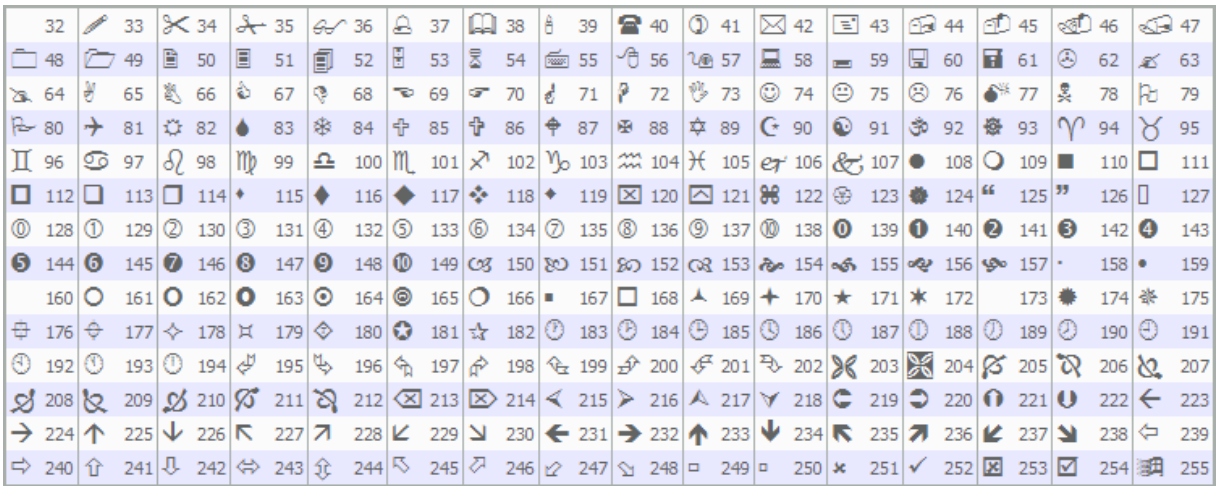
예:

```
//---- 지표 설정
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_type3   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_color2  clrRed
#property indicator_color3  clrLime
```

© 2000-2025, MetaQuotes Ltd.

Wingdings

OBJ_ARROW 개체와 함께 사용되는 윙딩 문자:



필요한 문자는 ObjectSetInteger() 함수를 사용하여 설정할 수 있습니다.

예:

```
void OnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);      // 종가 가져오기
//--- 가격을 가져온 경우
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);           // 화살표 생성
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);     // 화살표 코드 설정
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);         // 시간 설정
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// 가격 설정
      ChartRedraw(0);                                         // 화살표 지금 그리기
     }
   else
      Print("최신 마감 가격을 가져올 수 없습니다");
  }
```

© 2000-2025, MetaQuotes Ltd.

지표 상수

MQL5 언어로 작성된 프로그램에서 사용할 수 있는 37개의 사전 정의된 기술 지표가 있습니다. 또한 iCustom() 함수를 사용하여 사용자 지정 지표를 만들 수 있습니다. 필요한 모든 상수는 5개의 그룹으로 나뉩니다:

- 가격 상수 – 지표가 계산되는 가격 또는 수량 유형을 선택;
- 평활법 – 지표에 사용되는 내장 평활법;
- 지시선 - CopyBuffer()를 사용하여 지표 값에 액세스할 때 지표 버퍼의 식별자;
- 그리기 스타일 – 18가지 도면 유형 중 하나를 표시하고 선 그리기 스타일을 설정하는 데 사용;
- 사용자 지정 지표 속성은 사용자 지정 지표로 작업하기 위한 함수에 사용;
- 지표의 유형은 IndicatorCreate()를 사용하여 핸들을 작성할 때 기술적 지표 유형을 지정하는데 사용;
- 데이터 유형 식별자는 MqlParam 유형의 배열에서 IndicatorCreate() 함수로 전달되는 데이터 유형을 지정하는 데 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

가격 상수

기술 지표에는 계산에는 가격 값 및/또는 볼륨 값이 필요하며 이 값에 따라 계산이 수행됩니다. 계산을 위해 원하는 가격 기준을 지정하는 데 사용되는 ENUM_APPLIED_PRICE 열거형에는 7개의 미리 정의된 식별자가 있습니다.

ENUM_APPLIED_PRICE

ID	설명
PRICE_CLOSE	종가
PRICE_OPEN	시가
PRICE_HIGH	기간의 최대 가격
PRICE_LOW	기간의 최소 가격
PRICE_MEDIAN	중위 가격, (high + low)/2
PRICE_TYPICAL	일반 가격, (high + low + close)/3
PRICE_WEIGHTED	평균 가격, (high + low + close + close)/4

볼륨이 계산에 사용되는 경우 ENUM_APPLIED_VOLUME 열거형에서 두 값 중 하나를 지정해야 합니다.

ENUM_APPLIED_VOLUME

ID	설명
VOLUME_TICK	틱 볼륨
VOLUME_REAL	거래 볼륨

iStochastic() 기술적 지표는 다음 두 가지 방법으로 계산할 수 있습니다:

- 오직 종가만;
- 또는 고가 및 저가.

계산에 필요한 변형을 선택하려면 ENUM_STO_PRICE 열거값 중 하나를 지정합니다.

ENUM_STO_PRICE

ID	설명
STO_LOWHIGH	저가/고가를 기준으로 계산
STO_CLOSECLOSE	마감/종가 기준으로 계산

기술 지표가 ENUM_APPLIED_PRICE에 의해 설정된 유형의 가격 데이터를 사용하는 경우 모든 지표의 핸들(터미널에 내장되거나 사용자가 작성한)을 입력 가격 시리즈로 사용할 수 있습니다. 이 경우 지표의 0 버퍼 값이 계산에 사용됩니다. 이렇게 하면 다른 지표의 값을 사용하여 한 지표 값을 쉽게 작성할 수 있습니다. 사용자 지정 지표 핸들은 iCustom() 함수를 호출하여 생성됩니다.

예:

```
#property indicator_separate_window
#property indicator_buffers 2
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   2
//--- 입력 매개변수
input int      RSIperiod=14;         // RSI 계산 기간
input int      Smooth=8;             // 평활 기간 RSI
input ENUM_MA_METHOD meth=MODE_SMMA; // 평활법
//---- RSI 플롯
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- RSI_Smoothed 플롯
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- 지표 버퍼
double         RSIBuffer[];          // 여기에 RSI 값을 저장합니다
double         RSI_SmoothedBuffer[]; // 다음은 RSI 의 평활값입니다
int            RSIhandle;            // RSI 지표에 핸들 연결
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 지표 버퍼 매핑
   SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
   IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
   RSIhandle=iRSI(NULL,0,RSIperiod,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[]
                )

  {
//---  마지막 오류 값 재설정
   ResetLastError();
//--- RSIBuffer []에서 RSI 지표 데이터 가져오기
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(copied<=0)
     {
      Print("지표 RSI 값을 복사할 수 없습니다. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- RSI 값을 사용하여 평균값 지표 생성
   int RSI_MA_handle=iMA(NULL,0,Smooth,0,meth,RSIhandle);
   copied=CopyBuffer(RSI_MA_handle,0,0,rates_total,RSI_SmoothedBuffer);
   if(copied<=0)
     {
      Print("RSI 의 평활 지표를 복사할 수 없습니다. Error = ",
            GetLastError(),",  copied =",copied);
      return(0);
     }
//--- 다음 번 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

평활법

많은 기술적 지표는 다양한 가격 시리즈 평활법을 기반으로 합니다. 일부 표준 기술적 지표에는 평활화 유형을 입력 모수로 지정해야 합니다. 원하는 평활 유형을 지정하기 위해 ENUM_MA_METHOD 열거에 나열된 식별자가 사용됩니다.

ENUM_MA_METHOD

ID	설명
MODE_SMA	단순 평균
MODE_EMA	지수 평균
MODE_SMMA	평활 평균
MODE_LWMA	선형 가중 평균

예:

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- 이동 평균 핸들
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- MA의 핸들 가져오기
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

© 2000-2025, MetaQuotes Ltd.

지시선

일부 기술적 지표에는 차트에 여러 버퍼가 그려져 있습니다. 지표 버퍼 번호 지정은 0으로 시작합니다. CopyBuffer() 함수를 사용하여 지표 값을 이중 유형의 배열로 복사할 때 일부 지표의 경우 번호 대신 복사된 버퍼의 식별자를 나타낼 수 있습니다.

Identifiers of indicator lines permissible when copying values of iMACD(), iRVI() 및 iStochastic() 값을 복사할 때 허용되는 지시선 식별자.

정수	값	설명
MAIN_LINE	0	본선
SIGNAL_LINE	1	신호선

ADX() 및 ADXW() 값을 복사할 때 허용되는 지시선 식별자.

정수	값	설명
MAIN_LINE	0	본선
PLUSDI_LINE	1	선 +DI
MINUSDI_LINE	2	선 -DI

iBands() 값을 복사할 때 허용되는 지시선 식별자.

정수	값	설명
BASE_LINE	0	본선
UPPER_BAND	1	상한
LOWER_BAND	2	하한

iEnvelopes() 및 iFractals() 값을 복사할 때 허용되는 지시선 식별자.

정수	값	설명
UPPER_LINE	0	상부선
LOWER_LINE	1	하부선

iGator() 값을 복사할 때 허용되는 지시선 식별자

정수	값	설명
UPPER_HISTOGRAM	0	상부 히스토그램
LOWER_HISTOGRAM	2	하단 히스토그램

iAlligator() 값을 복사할 때 허용되는 지시선 식별자.

© 2000-2025, MetaQuotes Ltd.

정수	값	설명
GATORJAW_LINE	0	턱 선
GATORTEETH_LINE	1	톱니 선
GATORLIPS_LINE	2	입술 선

IIchimoku() 값을 복사할 때 허용되는 지시선 식별자.

정수	값	설명
TENKANSEN_LINE	0	전환선
KIJUNSEN_LINE	1	기준선
SENKOUSPANA_LINE	2	선행 스팬 A 선
SENKOUSPANB_LINE	3	선행 스팬 B 선
CHIKOUSPAN_LINE	4	후행 스팬 선

© 2000-2025, MetaQuotes Ltd.

그리기 스타일

사용자 지정 지표를 만들 때 기본 차트 창 또는 차트 하위 창에 표시되는 18가지 그래핑 플로팅 유형 중 하나를 지정할 수 있습니다. ENUM_DRAW_TYPE 열거에 지정됩니다.

하나의 사용자 지정 지표에서 모든지표 빌딩/도면 유형을 사용할 수 있습니다. 각 구성 유형에는 그리기에 필요한 데이터를 저장하기 위한 1~5 개의 글로벌 배열 사양이 필요합니다. 이러한 데이터 배열은 SetIndexBuffer() 함수를 사용하여 지표 버퍼로 바인딩되어야 합니다. 각 버퍼에 대해 ENUM_INDEXBUFFER_TYPE의 데이터 유형을 지정해야 합니다.

그리기 스타일에 따라 1~4개의 값 버퍼 (INDICATOR_DATA로 표시됨)가 필요할 수 있습니다. 스타일에 동적 색상 교체를 허용하는 경우(모든 COLOR가 포함됨), 하나의 색상 버퍼가 더 필요합니다(표시돈 유형 INDICATOR_COLOR_INDEX). 색상 버퍼는 항상 스타일에 해당하는 값 버퍼 뒤에 바인딩됩니다.

ENUM_DRAW_TYPE

ID	설명	Data buffers	Color buffers
DRAW_NONE	Not drawn	1	0
DRAW_LINE	선	1	0
DRAW_SECTION	섹션	1	0
DRAW_HISTOGRAM	0선으로부터의 히스토그램	1	0
DRAW_HISTOGRAM2	두 지표 버퍼의 히스토그램	2	0
DRAW_ARROW	화살표 그리기	1	0
DRAW_ZIGZAG	스타일 지그재그로 막대에서 수직 단면 허용	2	0
DRAW_FILLING	두 수준 사이의 색상 채우기	2	0
DRAW_BARS	일련의 막대로 표시	4	0

© 2000-2025, MetaQuotes Ltd.

ID	설명	Data buffers	Color buffers
DRAW_CANDLES	일련의 촛대로 표시	4	0
DRAW_COLOR_LINE	다중 색상 선	1	1
DRAW_COLOR_SECTION	다중 색상 섹션	1	1
DRAW_COLOR_HISTOGRAM	0선의 다중 색상 히스토그램	1	1
DRAW_COLOR_HISTOGRAM2	두 지표 버퍼의 다중 색상 히스토그램	2	1
DRAW_COLOR_ARROW	다중 색상 화살표 그리기	1	1
DRAW_COLOR_ZIGZAG	다중 색상 지그재그	2	1
DRAW_COLOR_BARS	다중 색상 막대	4	1
DRAW_COLOR_CANDLES	다중 색상 캔들스틱	4	1

ENUM_PLOT_PROPERTY에 나열된 선택한 그리기 유형 식별자의 화면표시를 세분화하기 위해 사용됩니다.

다음 함수의 경우: PlotIndexSetInteger() 및 PlotIndexGetInteger()

ENUM_PLOT_PROPERTY_INTEGER

ID	설명	속성 유형
PLOT_ARROW	스타일 DRAW_ARROW에 대한 화살표	uchar
PLOT_ARROW_SHIFT	스타일 DRAW_ARROW에	int

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
	대한 화살표의 수직 이동	
PLOT_DRAW_BEGIN	DataWindow에서 그려지지 않은 초기 막대 수 및 값	int
PLOT_DRAW_TYPE	그래픽 구성 유형	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	DataWindow에 구성 값 표시 기호	bool
PLOT_SHIFT	막대의 시간 축을 따라 표시된 지표 이동	int
PLOT_LINE_STYLE	선 그리기 스타일	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	그리기 선 굵기	int
PLOT_COLOR_INDEXES	색상의 숫자	int
PLOT_LINE_COLOR	그리기 색상을 포함하는 버퍼의 인덱스	color 제어자 = 색상 인덱스 수

다음 함수의 경우: PlotIndexSetDouble()

ENUM_PLOT_PROPERTY_DOUBLE

ID	설명	속성 유형
PLOT_EMPTY_VALUE	그리기가 없는 빈 플롯 값	double

다음 함수의 경우: PlotIndexSetString()

ENUM_PLOT_PROPERTY_STRING

ID	설명	속성 유형
PLOT_LABEL	DataWindow에 표시할 지표 그래픽 시리즈의 이름. 표시할 여러 지표 버퍼가 필요한 복잡한 그래픽 스타일로 작업할 경우 ","를 구분 기호로 사용하여 각 버퍼의 이름을 지정할 수 있습니다. 샘플 코드가	string

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
	DRAW_CANDLES 에 표시됩니다	

사용자 지정 지표에서 선을 그리는 데 5가지 스타일을 사용할 수 있습니다. 선 굵기 0 또는 1에 대해서만 유효합니다.

ENUM_LINE_STYLE

ID	설명
STYLE_SOLID	실선
STYLE_DASH	파선
STYLE_DOT	점선
STYLE_DASHDOT	일점 쇄선
STYLE_DASHDOTDOT	이점 쇄선

선 그리기 스타일 및 도면 유형을 설정하려면 PlotIndexSetInteger() 함수를 사용합니다. 피보나치 확장의 경우 ObjectSetInteger() 함수를 사용하여 레벨의 두께 및 그리기 스타일을 표시할 수 있습니다.

예:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- 지표 버퍼
double         MABuffer[];
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 배열을 인덱스가 0인 지표 버퍼에 바인딩합니다.
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- 선 그리기 설정
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- 선 스타일 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- 선 색상 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- 선 굵기 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- 선 레이블 설정
   PlotIndexSetString(0,PLOT_LABEL,"이동 평균");
//---
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 사용자 지정 지표 반복 함수                                        |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   for(int i=prev_calculated;i<rates_total;i++)
     {
      MABuffer[i]=close[i];
     }
//--- 다음 번 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

사용자 지정 지표 속성

사용자 지정 지표에 사용할 수 있는 지표 버퍼 수는 제한이 없습니다. 그러나 SetIndexBuffer() 함수를 사용하여 지표 버퍼로 지정된 각 배열에 대해 저장할 데이터 유형을 지정해야 합니다. 이 값은 ENUM_INDEXBUFFER_TYPE 열거값 중 하나일 수 있습니다.

ENUM_INDEXBUFFER_TYPE

ID	설명
INDICATOR_DATA	그릴 데이터
INDICATOR_COLOR_INDEX	색상
INDICATOR_CALCULATIONS	중간 계산을 위한 보조 버퍼

사용자 지정 지표에는 편리하게 표시할 수 있는 많은 설정이 있습니다. 이러한 설정은 IndicatorSetDouble(), IndicatorSetInteger() 및 IndicatorSetString() 함수를 사용하여 해당 지표 특성을 할당하여 수행됩니다. 지표 속성 식별자가 ENUM_CUSTOMIND_PROPERTY 열거에 나열됩니다.

ENUM_CUSTOMIND_PROPERTY_INTEGER

ID	설명	속성 유형
INDICATOR_DIGITS	지시자 값 도면의 정확도	int
INDICATOR_HEIGHT	지표 창의 고정 높이 (전처리기 명령 #property indicator_height)	int
INDICATOR_LEVELS	지표 창의 레벨의 수	int

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
INDICATOR_LEVELCOLOR	레벨 선의 색상	색상 제어자 = 레벨 숫자
INDICATOR_LEVELSTYLE	레벨 선의 스타일	ENUM_LINE_STYLE 제어자 = 레벨 숫자
INDICATOR_LEVELWIDTH	레벨 선의 두께	int 제어자 = 레벨 숫자
INDICATOR_FIXED_MINIMUM	지표창의 최소값 수정. 속성은 오직 IndicatorSetInteger() 함수로만 작성 가능합니다	bool
INDICATOR_FIXED_MAXIMUM	지표창의 최대값 수정. 속성은 오직 IndicatorSetInteger() 함수로만 작성	bool

© 2000-2025, MetaQuotes Ltd.

ID	설명	속성 유형
	가능합니다	

ENUM_CUSTOMIND_PROPERTY_DOUBLE

ID	설명	속성 유형
INDICATOR_MINIMUM	지표창의 최소값	double
INDICATOR_MAXIMUM	지표창의 최대값	double
INDICATOR_LEVELVALUE	레벨 값	double 제어자 = 레벨 숫자

ENUM_CUSTOMIND_PROPERTY_STRING

ID	설명	속성 유형
INDICATOR_SHORTNAME	short 지표 이름	string
INDICATOR_LEVELTEXT	레벨 설명	string 제어자 = 레벨 숫자

예:

```
//--- 지표 설정
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
#property indicator_type1   DRAW_LINE
#property indicator_type2   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_color2  clrRed
//--- 입력 매개변수
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- 지표 버퍼
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
```

© 2000-2025, MetaQuotes Ltd.

```
double LowesBuffer[];
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 지표 버퍼 매핑
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- 정확성 설정
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- 레벨 설정
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- 하위 창의 최대값 및 최소값 설정
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- 인덱스를 그릴 첫 번째 막대를 설정
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- 두 번째 선의 STYLE_DOT 설정
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- DataWindow 및 지표 하위 창 레이블
   IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")"
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- 도면 선을 빈 값으로 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- 초기화 완료
  }
```

© 2000-2025, MetaQuotes Ltd.

기술적 지표의 유형

추가로 해당 값에 액세스하기 위한 지표 핸들을 만드는 두 가지 방법이 있습니다. 첫 번째 방법은 기술적 지표목록에서 기능 이름을 직접 지정하는 것입니다. IndicatorCreate()를 사용하는 두 번째 방법은 ENUM_INDICATOR 열거에서 식별자를 할당하여 지표의 핸들을 균일하게 만드는 것입니다. 두 가지 핸들 생성 방법은 모두 동일하면서 MQL5에서 프로그램을 작성할 때 특정 경우에 가장 편리한 핸들 생성 방법을 사용할 수 있습니다.

IND_CUSTOM 유형의 지표를 만드는 경우, *유형* 필드(입력 매개변수 MqlParam 배열의 첫 번째 요소)에는 ENUM_DATATYPE의 TYPE_STRING 값이 있어야 하며 첫 번째 요소의 *string_value* 필드에는 사용자 지정 지표의 이름이 포함되어야 합니다.

ENUM_INDICATOR

식별자	지표
IND_AC	가속기 오실레이터
IND_AD	누적/분산
IND_ADX	평균 방향 지수
IND_ADXW	웰스 와일더의 ADX
IND_ALLIGATOR	앨리게이터
\IND_AMA	조정 이동 평균
IND_AO	어썸 오실레이터
IND_ATR	평균 참값 범위
IND_BANDS	불린저 밴즈®
IND_BEARS	베어즈 파워
IND_BULLS	불스 파워
IND_BWMFI	시장 촉진 지수
IND_CCI	상품 채널 인덱스
IND_CHAIKIN	차이킨 오실레이터
IND_CUSTOM	사용자 지정 지표
IND_DEMA	이중 지수 이동 평균
IND_DEMARKER	디마커(DeMarker)
IND_ENVELOPES	엔벨로프
IND_FORCE	포스 인덱스
IND_FRACTALS	프랙탈
IND_FRAMA	프랙탈 조정 이동 평균
IND_GATOR	게이터 오실레이터

© 2000-2025, MetaQuotes Ltd.

식별자	지표
IND_ICHIMOKU	일목균형표
IND_MA	이동 평균
IND_MACD	MACD
IND_MFI	자금 흐름 지수
IND_MOMENTUM	모멘텀
IND_OBV	누적 거래량 지표
IND_OSMA	OsMA
IND_RSI	상대 강도 지수
IND_RVI	상대 활력 지수
IND_SAR	파라볼릭 SAR
IND_STDDEV	표준 편차
IND_STOCHASTIC	스토캐스틱 오실레이터
IND_TEMA	삼중 지표 이동 평균
IND_TRIX	삼중 지표 이동 평균 오실레이터
IND_VIDYA	변수 인덱스 동적 평균
IND_VOLUMES	볼륨
IND_WPR	윌리엄스의 백분율 범위

© 2000-2025, MetaQuotes Ltd.

데이터 유형 식별자

IndicatorCreate() 함수를 사용하여 지표 핸들을 작성할 때 마지막 매개 변수로 MqlParam 유형의 배열을 지정해야 합니다. 따라서, 지표를 설명하는 MqlParam 구조는 특수 필드 *유형*을 포함합니다. 이 필드에서 배열의 특정 요소에서 전달되는 데이터 유형 (실수, 정수 또는 문자열 유형)에 대한 정보가 포함되어 있습니다. MqlParam 구조의 이 필드 값은 ENUM_DATATYPE 값 중 하나일 수 있습니다.

ENUM_DATATYPE

식별자	데이터 유형
TYPE_BOOL	bool
TYPE_CHAR	char
TYPE_UCHAR	uchar
TYPE_SHORT	short
TYPE_USHORT	ushort
TYPE_COLOR	색상
TYPE_INT	int
TYPE_UINT	uint
TYPE_DATETIME	datetime
TYPE_LONG	long
TYPE_ULONG	ulong
TYPE_FLOAT	float
TYPE_DOUBLE	double
TYPE_STRING	string

배열의 각 요소는 작성된 기술적 지표의 해당 입력 매개변수를 설명하므로 배열의 요소 유형 및 순서는 설명에 따라 엄격하게 유지되어야 합니다.

© 2000-2025, MetaQuotes Ltd.

환경 상태

mql5-프로그램의 현재 런타임 환경을 설명하는 상수는 다음과 같이 그룹으로 나뉩니다:

- 클라이언트 터미널 속성 – 클라이언트 너미널에 대한 정보;
- 실행된 MQL5-프로그램 속성 – 실행을 제어하는 데 도움이 되는 mql5 프로그램 속성;
- 기호 속성 – 기호 정보 가져오기;
- 계정 속성 – 현재 계정에 대한 정보;
- 통계 검증 – 엑스퍼트 어드바이저 검증 결과.

© 2000-2025, MetaQuotes Ltd.

클라이언트 터미널 속성

클라이언트 터미널에 대한 정보는 두 가지 기능으로 얻을 수 있습니다: TerminalInfoInteger() 및 TerminalInfoString(). 매개변수의 경우 이러한 함수는 각각 ENUM_TERMINAL_INFO_INTEGER 및 ENUM_TERMINAL_INFO_STRING 값을 허용합니다.

ENUM_TERMINAL_INFO_INTEGER

식별자	설명	유형
TERMINAL_BUILD	클라이언트 터미널 빌드 번호	int
TERMINAL_COMMUNITY_ACCOUNT	이 플래그는 터미널의 MQL5.community 권한 부여가 있음을 나타냅니다	bool
TERMINAL_COMMUNITY_CONNECTION	MQL5.community로 연결	bool
TERMINAL_CONNECTED	거래 서버 연결	bool
TERMINAL_DLLS_ALLOWED	DLL 사용 권한	bool
TERMINAL_TRADE_ALLOWED	거래 권한	bool
TERMINAL_EMAIL_ENABLED	터미널 설정에 지정된 SMTP-서버 및 사용자명을 사용하여 이메일을 보낼 수 있는 권한	bool
TERMINAL_FTP_ENABLED	터미널 설정에 지정된 FTP-서버 및 사용자명을 사용하여 보고서를 보낼 수 있는 권한	bool
TERMINAL_NOTIFICATIONS_ENABLED	스마트폰에 알림 전송 권한	bool
TERMINAL_MAXBARS	차트에서 최대 막대 카운트	int
TERMINAL_MQID	이 플래그는 푸쉬 알림에 대한 MetaQuotes ID 데이터의 존재를 나타냅니다	bool
TERMINAL_CODEPAGE	클라이언트 터미널에 설치된언어의 코드 페이지 수	int
TERMINAL_CPU_CORES	시스템의 CPU 코어 수	int
TERMINAL_DISK_SPACE	터미널(에이전트)의 MQL5₩Files 폴더에 사용할 수 있는 디스크 공간(MB)	int
TERMINAL_MEMORY_PHYSICAL	시스템의 물리적 메모리(MB)	int
TERMINAL_MEMORY_TOTAL	터미널 에이전트 프로세스에 사용할 수 있는 메모리(MB)	int
TERMINAL_MEMORY_AVAILABLE	터미널(에이전트) 프로세스의 여유 메모리(MB)	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
TERMINAL_MEMORY_USED	터미널(에이전트)에서 사용하는 메모리(MB)	int
TERMINAL_X64	"64비트 터미널" 표시	bool
TERMINAL_OPENCL_SUPPORT	0x00010002 = 1.2 형식의 지원되는 OpenCL 버전. "0"은 OpenCL이 지원되지 않음을 의미합니다	int
TERMINAL_SCREEN_DPI	화면의 정보 디스플레이 해상도는 인치당 한 줄에 있는 도트 수(DPI)로 측정됩니다. 매개변수 값을 알고 있으면 해상도 특성이 다른 모니터에서 동일하게 보이도록 그래픽 개체의 크기를 설정할 수 있습니다.	int
TERMINAL_SCREEN_LEFT	가상 화면의 왼쪽 좌표. 가상 화면은 모든 모니터를 대신하는 사각형입니다. 시스템에 두 대의 모니터가 우측에서 좌측으로 정렬되어 있으면 가상 화면의 좌측 좌표가 두 대의 모니터 경계이 있을 수 있습니다.	int
TERMINAL_SCREEN_TOP	가상 화면의 상단 좌표	int
TERMINAL_SCREEN_WIDTH	터미널 너비	int
TERMINAL_SCREEN_HEIGHT	터미널 높이	int
TERMINAL_LEFT	가상 화면에 상대적인 터미널의 좌측 좌표	int
TERMINAL_TOP	가상 화면에 상대적인 터미널의 상단 좌표	int
TERMINAL_RIGHT	가상 화면에 상대적인 터미널의 우측 좌표	int
TERMINAL_BOTTOM	가상 화면에 상대적인 터미널의 하단 좌표	int
TERMINAL_PING_LAST	거래 서버에 대한 ping의 알려진 마지막 값(마이크로초). 1초는 100만 마이크로초 입니다.	int
TERMINAL_VPS	터미널이 MetaTrader 가상 호스팅 서버 (MetaTrader VPS)에서 시작되었음을 나타냅니다	bool
키 식별자	설명	
TERMINAL_KEYSTATE_LEFT	"왼쪽 화살표" 키의 상태	int
TERMINAL_KEYSTATE_UP	"위쪽 화살표" 키의 상태	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
TERMINAL_KEYSTATE_RIGHT	"오른쪽 화살표" 키의 상태	int
TERMINAL_KEYSTATE_DOWN	"아래쪽 화살표" 키의 상태	int
TERMINAL_KEYSTATE_SHIFT	"시프트" 키의 상태	int
TERMINAL_KEYSTATE_CONTROL	"컨트롤" 키의 상태	int
TERMINAL_KEYSTATE_MENU	"윈도우" 키의 상태	int
TERMINAL_KEYSTATE_CAPSLOCK	"캡스락" 키의 상태	int
TERMINAL_KEYSTATE_NUMLOCK	"넘버락" 키의 상태	int
TERMINAL_KEYSTATE_SCRLOCK	"스크롤락" 키의 상태	int
TERMINAL_KEYSTATE_ENTER	"엔터" 키의 상태	int
TERMINAL_KEYSTATE_INSERT	"인서트" 키의 상태	int
TERMINAL_KEYSTATE_DELETE	"딜리트" 키의 상태	int
TERMINAL_KEYSTATE_HOME	"홈" 키의 상태	int
TERMINAL_KEYSTATE_END	"엔드" 키의 상태	int
TERMINAL_KEYSTATE_TAB	"탭" 키의 상태	int
TERMINAL_KEYSTATE_PAGEUP	"페이지업" 키의 상태	int
TERMINAL_KEYSTATE_PAGEDOWN	"페이지다운" 키의 상태	int
TERMINAL_KEYSTATE_ESCAPE	"esc" 키의 상태	int

TerminalInfoInteger(TERMINAL_KEYSTATE_XXX) 호출 시 MSDN의 GetKeyState()함수와 동일한 키 상태 코드가 반환됩니다.

예제 – 스케일 팩터 계산:

```
//--- 화면에 1.5인치 폭 버튼 만들기
int screen_dpi = TerminalInfoInteger(TERMINAL_SCREEN_DPI); // 사용자 모니터의 DPI 찾기
int base_width = 144;                                      // 화면의 기본 너비는 DPI=96
int width      = (button_width * screen_dpi) / 96;         // 사용자 모니터의 버튼 폭 계
...

예제 - 스케일 팩터 계산:
int scale_factor=(TerminalInfoInteger(TERMINAL_SCREEN_DPI) * 100) / 96;
//--- 스케일 팩터 사용
width=(base_width * scale_factor) / 100;
```

위의 예에서 그래픽 리소스는 해상도 특성이 다른 모니터에서 동일하게 표시됩니다. 제어 요소(버튼, 대화상자 창 등)의 크기는 개인 설정에 해당합니다.

© 2000-2025, MetaQuotes Ltd.

ENUM_TERMINAL_INFO_DOUBLE

식별자	설명	유형
TERMINAL_COMMUNITY_BALANCE	MQL5.community의 잔액	double
TERMINAL_RETRANSMISSION	지정된 컴퓨터에서 실행 중인 모든 애플리케이션 및 서비스에 대한 TCP/IP 프로토콜 재연결 네트워크 패킷 백분율입니다. 패킷 손실은 가장 빠르고 올바르게 구성된 네트워크에서도 발생합니다. 이 경우 수신자와 보낸 사람 사이에 패킷 전달이 확인되지 않으므로 손실된 패킷이 재전송됩니다. 시스템 및 백그라운드 작업을 포함한 전체 네트워크 작업에 대해 백분율이 계산되므로 특정 터미널과 거래 서버간의 연결 품질을 나타내는 것은 아닙니다. TERMINAL_RETRANSMISSION 값은 분당 한 번씩 운영 시스템에 요청됩니다. 터미널 자체는 이 값을 계산하지 않습니다.	double

파일 작업은 두개의 디렉토리에서만 수행할 수 있으며 TERMINAL_DATA_PATH 및 TERMINAL_COMMONDATA_PATH 속성 요청을 사용하여 해당 경로를 가져올 수 있습니다.

ENUM_TERMINAL_INFO_STRING

식별자	설명	유형
TERMINAL_LANGUAGE	터미너리 언어	string
TERMINAL_COMPANY	회사 이름	string
TERMINAL_NAME	터미널 이름	string
TERMINAL_PATH	터미널이 시작되는 폴더	string
TERMINAL_DATA_PATH	터미널 데이터가 저장되는 폴더	string
TERMINAL_COMMONDATA_PATH	컴퓨터에 설치된 모든 터미널의 공통 경로	string
TERMINAL_CPU_NAME	CPU 명	string
TERMINAL_CPU_ARCHITECTURE	CPU 아키텍쳐	string
TERMINAL_OS_VERSION	사용자 운영체제 명	string

© 2000-2025, MetaQuotes Ltd.

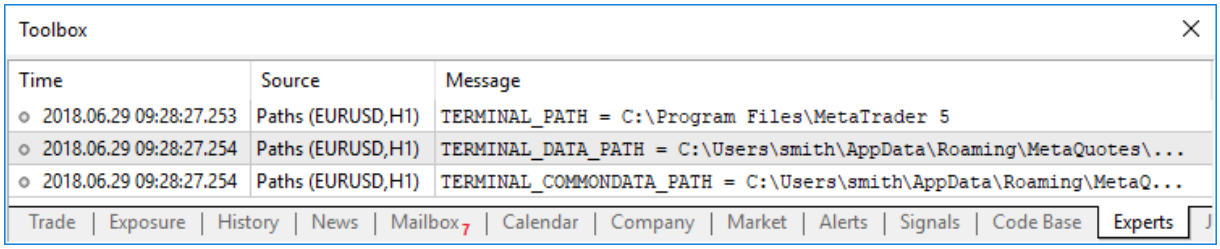
식별자	설명	유형
TERMINAL_COLORTHEME_NAME	터미널 색 구성표; 가능한 값 : 밝은 색과 어두운색	string

TERMINAL_PATH, TERMINAL_DATA_PATH 및 TERMINAL_COMMONDATA_PATH 매개변수의 속성에 저장된 경로를 더 잘 알기 위해 컴퓨터에 설치된 클라이언트 터미널의 현재 복사본에 대해 이 값을 반환하는 스크립트를 실행하는 것이 좋습니다

예: 스크립트가 클라이언트 터미널 경로에 대한 정보를 반환합니다

```
//+------------------------------------------------------------------+
//|                                          Check_TerminalPaths.mq5 |
//|                        Copyright 2009, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
   Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
   Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
  }
```

엑스퍼트 저널에서 스크립트를 실행하면 다음과 같은 메시지가 표시됩니다:



터미널 색 구성표 정보

터미널은 두 가지 색 구성표 밝은색 (기본값)과 어두운 색을 지원합니다. 그래픽 사용자 인터페이스를 사용하여 사용자 정의 응용 프로그램을 개발할 때 프로그래머는 현재 터미널의 색 구성표를 고려해야합니다. 응용 프로그램에 사용되는 시각적 구성 요소는 사용자 경험을 향상시키고 시각적 일관성을 유지하기 위해 동적으로 적응될 수 있어야합니다.

이를 지원하기 위해 터미널 색 구성표를 감지하기위한 함수를 제공합니다.

© 2000-2025, MetaQuotes Ltd.

- The TERMINAL_COLORTHEME_NAME value from the ENUM_TERMINAL_INFO_STRING enumeration allows you to retrieve the name of the current color scheme using the TerminalInfoString function. 가능한 값: 밝은색과 어두운 색
- Use the THEME_COLOR_* values from the ENUM_TERMINAL_INFO_INTEGER enumeration to retrieve the colors of specific UI elements through the TerminalInfoInteger function.

식별자	설명	속성 타입
THEME_COLOR_WINDOW	창 백그라운드	색상
THEME_COLOR_WINDOWTEXT	창 텍스트	색상
THEME_COLOR_BTNTEXT	버튼 텍스트	색상
THEME_COLOR_GRAYTEXT	비활성(사용 불가한) 텍스트	색상
THEME_COLOR_INFOTEXT	툴팁 텍스트	색상
THEME_COLOR_INFOBK	툴팁 백그라운드	색상
THEME_COLOR_3DFACE	3D 요소의 프론트	색상
THEME_COLOR_3DLIGHT	3D 요소의 밝은 사이드	색상
THEME_COLOR_3DSHADOW	3D 요소의 그림자 사이드	색상
THEME_COLOR_3DDKSHADOW	3D 요소의 어두운 그림자	색상
THEME_COLOR_3DHILIGHT	3D 요소의 강조	색상
THEME_COLOR_HIGHLIGHT	선택된 요소의 백그라운드	색상
THEME_COLOR_HIGHLIGHTTEXT	선택된 요소의 텍스트	색상
THEME_COLOR_BTNFACE	버튼 프론트 페이스	색상
THEME_COLOR_BTNHILIGHT	버튼 강조	색상
THEME_COLOR_BTNSHADOW	버튼 그림자	색상
THEME_COLOR_MENU	메뉴 백그라운드	색상
THEME_COLOR_MENUBAR	메뉴바 백그라운드	색상
THEME_COLOR_MENUTEXT	메뉴 텍스트	색상
THEME_COLOR_MENUHILIGHT	선택된 메뉴 아이템의 강조	색상
THEME_COLOR_ACTIVECAPTION	활성화 창 타이틀	색상
THEME_COLOR_INACTIVECAPTION	비활성 창 타이틀	색상
THEME_COLOR_GRADIENTINACTIVECAPTION	비활성 창 타이틀의 경사	색상
THEME_COLOR_CAPTIONTEXT	창 타이틀 텍스트	색상
THEME_COLOR_INACTIVECAPTIONTEXT	비활성 창 타이틀 텍스트	색상
THEME_COLOR_HOTTEXT	하이퍼링크 또는 활성 요소	색상

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 타입
THEME_COLOR_NONE	선택되지 않은 색상	색상
THEME_COLOR_SEPARATOR	구분자	색상
THEME_COLOR_SCROLLBACK	스크롤바	색상
THEME_COLOR_LINE1	저널의 홀수 줄의 백그라운드 색	색상
THEME_COLOR_LINE2	저널의 짝수 줄의 백그라운드 색	색상
THEME_COLOR_GRID	저널의 그리드 색	색상
THEME_COLOR_SUMMARY	저널 요약 행의 백그라운드 색	색상
THEME_COLOR_ERROR	에러 메세지 텍스트 색	색상
THEME_COLOR_INVALID	잘못된 값 텍스트 색	색상
THEME_COLOR_NEGATIVE	음수값 색	색상
THEME_COLOR_POSITIVE	양수값 색	색상
THEME_COLOR_LINK	링크 색	색상
THEME_COLOR_LINKHOVER	링크 호버 색	색상
THEME_COLOR_LINKTESTER	이전 테스트/최적화 실행 캐시된 결과 의 링크 색	색상
THEME_COLOR_TEXTUP	"버튼 놓은" 상태	색상
THEME_COLOR_TEXTDOWN	"버튼 눌려진" 상태	색상
THEME_COLOR_BACKUP	호가가 증가 할 때 "매수"및 "매도"버튼의 색	색상
THEME_COLOR_BACKDOWN	호가가 줄어들 때 "매수"및 "매도"버튼의 색	색상
THEME_COLOR_CLOSE	"청산" 동작 버튼 색	색상
THEME_COLOR_BUY	"매수" 동작 버튼 색	색상
THEME_COLOR_SELL	"매도" 동작 버튼 색	색상
THEME_COLOR_DEPOSIT	"입금" 버튼 색	색상
THEME_COLOR_WITHDRAWAL	"인출" 버튼 색	색상
THEME_COLOR_BID	매도 호가 라인 색	색상
THEME_COLOR_ASK	매수 호가 라인 색	색상
THEME_COLOR_STOPS	스탑 라인 색	색상
THEME_COLOR_STOPS_RED	수익이 마이너스일 때 스탑로스 값 하이라이트(트레이드 탭)	색상
THEME_COLOR_STOPS_GREEN	수익이 플러스일 때 스탑로스 값 하이라이트(트레이드 탭)	색상

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 타입
THEME_COLOR_CONFIRM	주문 제출 창에서 "거절" 버튼 색	색상
THEME_COLOR_REQUOTE	주문 제출 창에서 "주문 재요청"버튼 색상	색상
THEME_COLOR_REJECT	주문 제출 창에서 "거절" 버튼 색	색상
THEME_COLOR_NOTIFICATION	주문 제출 창에서 서버로부터의 변경 알림의 색	색상
THEME_COLOR_RATING	학습 시스템의 등급 바 색	색상
THEME_COLOR_BOOK_BUY	시장 심도에서 매수 수준의 백그라운드 색	색상
THEME_COLOR_BOOK_SELL	시장 심도에서 매도 레벨의 백그라운드 색	색상
THEME_COLOR_BOOK_LAST	시장 심도에서 마지막 거래의 색	색상
THEME_COLOR_BOOK_STOP	시장 심도에서 스탑로스 레벨의 색	색상
THEME_COLOR_BOOK_SPREAD	시장 심도에서 스프레드 안에서의 레벨의 색	색상
THEME_COLOR_TICKS_BID	틱차트의 주문 제출 창에서 매도 호가 라인 색	색상
THEME_COLOR_TICKS_ASK	틱 차트의 주문 제출 창에서 매수 호가 라인 색	색상
THEME_COLOR_TICKS_LAST	틱차트의 주문 제출 창에서 마지막 줄 색	색상
THEME_COLOR_TICKS_CROSS	틱 차트의 주문 제출 창에서 크로스헤어 색	색상
THEME_COLOR_TICKS_SL	틱 차트에의 주문 제출 창에서 스탑로스라인 색	색상
THEME_COLOR_TICKS_TP	틱 차트에의 주문 제출 창에서 익절 라인 색	색상
THEME_COLOR_TESTER_START	테스트/최적화에서 "시작"버튼 색	색상
THEME_COLOR_TESTER_STOP	테스트/최적화에서 "중지" 버튼 색	색상
THEME_COLOR_TESTER_START_FRAME	테스트/최적화에서 "시작" 버튼의 경계선의 색	색상
THEME_COLOR_TESTER_STOP_FRAME	테스트/최적화에서 "중지" 버튼의 경계선의 색	색상
THEME_COLOR_TESTER_PROGRESS	테스트/최적화에서 진행상태 바의 색	색상

© 2000-2025, MetaQuotes Ltd.

식별자	설명	속성 타입
THEME_COLOR_TESTER_BALANCE	전략 테스터에서 잔고 라인의 색	색상
THEME_COLOR_TESTER_EQUITY	전략 테스터에서 예탁자산 평가 총액 라인의 색	색상
THEME_COLOR_TESTER_MARGIN	전략 테스터에서 입금 그래프 색	색상
THEME_COLOR_PROFILER_CALL	프로파일 링 중 요청 코드 라인의 색	색상
THEME_COLOR_PROFILER_CALLSEL	프로파일 링 중 요청 선택된 코드 라인의 색	색상
THEME_COLOR_PROFILER_LINE	프로파일러 저널에서 라인 색	색상
THEME_COLOR_PROFILER_LINESEL	프로파일러 저널에서 선택된 라인 색	색상

© 2000-2025, MetaQuotes Ltd.

MQL5 프로그램 속성 실행

현재 실행중인 mql5 프로그램에 대한 정보를 얻으려면 ENUM_MQL_INFO_INTEGER 및 ENUM_MQL_INFO_STRING의 상수가 사용됩니다.

다음 함수의 경우: MQLInfoInteger

ENUM_MQL_INFO_INTEGER

식별자	설명	유형
MQL_HANDLES_USED	현재 활성 개체 핸들 수. 여기에는 동적(new를 통해 생성됨) 및 비동적 개체, 전역/로컬 변수 또는 클래스 멤버가 모두 포함됩니다. 프로그램이 사용하는 핸들이 많을수록 더 많은 리소스를 소비합니다.	int
MQL_MEMORY_LIMIT	MQL5 프로그램에 대해 가능한 최대 동적 메모리 양(MB)	int
MQL_MEMORY_USED	MQL5 프로그램에서 사용하는 메모리 (MB)	int
MQL_PROGRAM_TYPE	MQL5 프로그램의 유형	ENUM_PROGRAM_TYPE
MQL_DLLS_ALLOWED	지정된 실행 프로그램에 대해 DLL을 사용할 수 있는 권한	bool
MQL_TRADE_ALLOWED	거래 권한 : 지정된 실행 프로그램에 대해	bool
MQL_SIGNALS_ALLOWED	시그널 수정 권한: 지정된 실행 프로그램에 대해	bool
MQL_DEBUG	프로그램이 디버깅 모드에서 실행 중임을 나타냄	bool
MQL_PROFILER	프로그램이 코드 프로파일링 모드	bool

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	에서 실행 중임을 나타냄	
MQL_TESTER	프로그램이 테스터에서 실행 중임을 나타냄	bool
MQL_FORWARD	프로그램이 전진 테스트 프로세스에서 실행 중임을 나타냄	bool
MQL_OPTIMIZATION	프로그램이 최적화 모드에서 실행 중임을 나타냄	bool
MQL_VISUAL_MODE	그로그램이 시각 테스트 모드에서 실행 중임을 나타냄	bool
MQL_FRAME_MODE	엑스퍼트 어드바이저가 수집 최적화 결과 프레임 모드에서 실행 중임을 나타냄	bool
MQL_LICENSE_TYPE	EX5 모듈의 라이센스 유형. 라이센스는 MQLInfoInteger(MQL_LICENSE_TYPE)를 사용하여 요청이 이루어진 EX5 모듈을 참조합니다	ENUM_LICENSE_TYPE
MQL_STARTED_FROM_CONFIG	configuration file의 시작 섹션에서 스크립트/EA가 시작된 경우 true를 반환합니다. 이는 이 스크립트/EA가 터미널이 실행되는 구성 파일에 등록되었음을 의미합니다	bool

다음 함수의 경우: MQLInfoString

ENUM_MQL_INFO_STRING

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
MQL_PROGRAM_NAME	실행 중인 mql5 프로그램 이름	string
MQL5_PROGRAM_PATH	경로: 지정된 실행 프로그램에 대해	string

실행 중인 프로그램 유형에 대한 내용은 ENUM_PROGRAM_TYPE 값을 사용합니다.

ENUM_PROGRAM_TYPE

식별자	설명
PROGRAM_SCRIPT	스크립트
PROGRAM_EXPERT	엑스퍼트
PROGRAM_INDICATOR	지표
PROGRAM_SERVICE	서비스

ENUM_LICENSE_TYPE

식별자	설명
LICENSE_FREE	무료 무제한 버전
LICENSE_DEMO	마켓에서 제공하는 유료 제품 평가판. 전략 테스터에서만 작동합니다
LICENSE_FULL	구입한 라이선스 버전은 최소 5개 이상의 활성화를 허용합니다. 활성화 횟수는 판매자에 의해 지정됩니다. 판매자의 허용된 활성화 수를 늘릴 수 있습니다
LICENSE_TIME	기간 제한이 있는 라이선스 버전

예:

```
ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)MQLInfoInteger(MQL_PROGRAM_TYPE);
switch(mql_program)
  {
   case PROGRAM_SCRIPT:
     {
      Print(__FILE__+" is script");
      break;
     }
   case PROGRAM_EXPERT:
     {
      Print(__FILE__+" is Expert Advisor");
      break;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   case PROGRAM_INDICATOR:
      {
       Print(__FILE__+" is custom indicator");
       break;
      }
   default:Print("MQL5 type value is ",mql_program);
//---
Print("MQLInfoInteger(MQL_MEMORY_LIMIT)=", MQLInfoInteger(MQL_MEMORY_LIMIT), " MB")
Print("MQLInfoInteger(MQL_MEMORY_USED)=", MQLInfoInteger(MQL_MEMORY_USED), " MB");
Print("MQLInfoInteger(MQL_HANDLES_USED)=", MQLInfoInteger(MQL_HANDLES_USED), " hand
```

© 2000-2025, MetaQuotes Ltd.

기호 속성

현재 시장 정보를 얻기 위해 몇 가지 기능이 있습니다: SymbolInfoInteger(), SymbolInfoDouble() 및 SymbolInfoString(). 첫 번째 매개변수는 기호 이름이고 두 번째 매개변수 값은 ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE 및 ENUM_SYMBOL_INFO_STRING의 식별자 중 하나일 수 있습니다.

다음 함수의 경우: SymbolInfoInteger()

ENUM_SYMBOL_INFO_INTEGER

식별자	설명	유형
SYMBOL_SECTOR	자산이 속한 경제 부문	ENUM_SYMBOL_SECTOR
SYMBOL_INDUSTRY	기호가 속한 산업 또는 경제 분기	ENUM_SYMBOL_INDUSTRY
SYMBOL_CUSTOM	사용자 지정 기호이며, 마켓 워치(MarketWatch) 및/또는 외부 데이터 소스의 다른 기호를 기반으로 합성	bool

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	적으로 작성되었습니다.	
SYMBOL_BACKGROUND_COLOR	마켓워치(MarketWatch)의 기호에 사용된 배경색	색상
SYMBOL_CHART_MODE	기호 막대를 생성하는 데 사용되는 가격 유형(예: 입찰 또는 마지막)	ENUM_SYMBOL_CHART_MODE
SYMBOL_EXIST	이 이름을 가진 기호가 있습니다	bool
SYMBOL_SELECT	마켓워치(MarketWat	bool

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	ch) 에서 기호가 선택됨	
SYMBOL_VISIBLE	마켓워치 (MarketWatch) 에 기호가 표시됨. 일부 기호 (대부분은 마진 요구사항이나 예금통화 수익 계산에 필요한 교차요금) 는 자동으로 선택되지만 마켓워치에서는 표시되지	bool

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	않을 수 있습니다. 이러한 기호를 표시하려면 명시적으로 선택해야 합니다.	
SYMBOL_SESSION_DEALS	현재 세션의 거래 수	long
SYMBOL_SESSION_BUY_ORDERS	현재 매수 주문 수	long
SYMBOL_SESSION_SELL_ORDERS	현재 매도 주문 수	long
SYMBOL_VOLUME	마지막 거래 볼륨	long
SYMBOL_VOLUMEHIGH	최대 일 볼륨	long
SYMBOL_VOLUMELOW	최소 일 볼륨	long
SYMBOL_TIME	마지막 견적 시간	datetime

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
SYMBOL_TIME_MSC	1970.01.01 이후 마지막 견적 시간 (밀리초)	long
SYMBOL_DIGITS	소수점 뒤의 자릿수	int
SYMBOL_SPREAD_FLOAT	플로팅 스프레드의 표시	bool
SYMBOL_SPREAD	스프레드 값 (포인트)	int
SYMBOL_TICKS_BOOKDEPTH	Depth of Market 에 나타난 요청 최대 수. 요청 대기열이 없는 기호의 경우 값은	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	0입니다.	
SYMBOL_TRADE_CALC_MODE	계약 가격 계산 모드	ENUM_SYMBOL_CALC_MODE
SYMBOL_TRADE_MODE	주문 실행 유형	ENUM_SYMBOL_TRADE_MODE
SYMBOL_START_TIME	기호 거래가 시작된 날짜 (일반적으로 선물에서 사용됨)	datetime
SYMBOL_EXPIRATION_TIME	기호 거래 종료 날짜 (일반적으로 선물에서 사용됨)	datetime
SYMBOL_TRADE_STOPS_LEVEL	현재 마감 가격에서 주문 중지까지의 포인트 최소 주의	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
SYMBOL_TRADE_FREEZE_LEVEL	포인트에서 거래 작업을 동결할 수 있는 거리	int
SYMBOL_TRADE_EXEMODE	거래 실행 모드	ENUM_SYMBOL_TRADE_EXECUTION
SYMBOL_SWAP_MODE	스왑 계산 모델	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	3일 스왑 롤오버 비용 청구 요일	ENUM_DAY_OF_WEEK
SYMBOL_MARGIN_HEDGED_USE_LEG	더 큰 leg 를 사용하여 위험 회피 마진 계산 (매수 또는 매도)	bool
SYMBOL_EXPIRATION_MODE	Flags of allowed order expi	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	ration modes	
SYMBOL_FILLING_MODE	허용된 주문 작성 모드 플래그	int
SYMBOL_ORDER_MODE	허용된 주문 유형 플래그	int
SYMBOL_ORDER_GTC_MODE	SYMBOL_EXPIRATION_MODE=SYMBOL_EXPIRATION_GTC인 경우 손실 중지 및 이익 취득 주문 접수 만료 (취소할 때까지	ENUM_SYMBOL_ORDER_GTC_MODE

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	유효)	
SYMBOL_OPTION_MODE	옵션 유형	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	옵션 우측 (콜/풋)	ENUM_SYMBOL_OPTION_RIGHT

다음 함수의 경우: SymbolInfoDouble()

ENUM_SYMBOL_INFO_DOUBLE

식별자	설명	유형
SYMBOL_BID	Bid – 최고 매도 제안	double
SYMBOL_BIDHIGH	오늘 최대 입찰	double
SYMBOL_BIDLOW	오늘 최소 입찰	double
SYMBOL_ASK	Ask – 최고 매수 제안	double
SYMBOL_ASKHIGH	오늘 최고 호가	double
SYMBOL_ASKLOW	오늘 최저 호가	double
SYMBOL_LAST	마지막 거래 가격	double
SYMBOL_LASTHIGH	오늘 최대 마지막 가격	double
SYMBOL_LASTLOW	오늘 최소 마지막 가격	double
SYMBOL_VOLUME_REAL	마지막 거래 볼륨	double
SYMBOL_VOLUMEHIGH_REAL	오늘 최대 볼륨	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
SYMBOL_VOLUMELOW_REAL	오늘 최소 볼륨	double
SYMBOL_OPTION_STRIKE	옵션 가격. 옵션 매수인이 기초자산을 매수(콜 옵션)하거나 매도(풋 옵션)할 수 있는 가격과 옵션 매도인은 기초자산을 적절한 금액으로 매도하거나 매수할 의무가 있습니다.	double
SYMBOL_POINT	기호 포인트 값	double
SYMBOL_TRADE_TICK_VALUE	SYMBOL_TRADE_TICK_VALUE_PROFIT의 값	double
SYMBOL_TRADE_TICK_VALUE_PROFIT	수익성 포지션에 대해 계산된 틱 가격	double
SYMBOL_TRADE_TICK_VALUE_LOSS	손실 포지션에 대해 계산된 틱 가격	double
SYMBOL_TRADE_TICK_SIZE	최소 가격 변동	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
SYMBOL_TRADE_CONTRACT_SIZE	거래 계약 크기	double
SYMBOL_TRADE_ACCRUED_INTEREST	미수 이자 - 누적 쿠폰 이자, 즉 쿠폰 채권 발행일수 또는 마지막 쿠폰 이자 지급일수에 비례하여 계산된 쿠폰 이자의 일부	double
SYMBOL_TRADE_FACE_VALUE	액면가 - 발행자가 설정한 최초 채권 가치	double
SYMBOL_TRADE_LIQUIDITY_RATE	유동성 요율은 마진에 사용할 수 있는 자산의 몫입니다.	double
SYMBOL_VOLUME_MIN	거래의 최소 볼륨	double
SYMBOL_VOLUME_MAX	거래의 최대 볼륨	double
SYMBOL_VOLUME_STEP	거래 실행의 최소 볼륨 변경 단계	double
SYMBOL_VOLUME_LIMIT	기호에 대해 한	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	방향(매수 또는 매도)으로 보류 중인 주문과 열린 포지션의 최대 허용 총량입니다. 예를 들어, 5로트 한도로 5로트 한정 오픈 매수 포지션을 갖게 되며, 5로트의 미결 매도 주문 제한을 할 수 있습니다. 그러나 이 경우 한 방향으로 총 물량이 한도를 초과하므로 보류 중인 구매 한도를 지정하거나 5로트 이상의 물량으로 판매 한도를 지정할 수 없습니다.	
SYMBOL_SWAP_LONG	장기 스왑 값	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
SYMBOL_SWAP_SHORT	단기 스왑 값	double
SYMBOL_SWAP_SUNDAY	SUNDAY에서 다음 날로 이월된 오버나잇 포지션에 대한 스왑 계산 비율 (SYMBOL_SWAP_LONG 또는 SYMBOL_SWAP_SHORT) 입니다. 다음 값이 지원됩니다: • 0 - 스왑 비용 없음 • 1 - 싱글 스왑 • 3 - 3중 스왑	double
SYMBOL_SWAP_MONDAY	월요일에서 화요일로 롤오버	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	된 오버나잇 포지션에 대한 스왑 계산 비율 (SYMBOL_SWAP_LONG 또는 SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_TUESDAY	화요일에서 수요일로 롤오버된 오버나잇 포지션에 대한 스왑 계산 비율 (SYMBOL_SWAP_LONG 또는 SYMBOL_SWAP_SHORT)	double
SYMBOL_SWAP_WEDNESDAY	화요일에서 수요일로 롤오버된 오버나잇 포지션에 대한 스왑 계산 비율 (SYMBOL_SWAP_LONG 또는 SYMBOL_SWAP_SHORT)	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
SYMBOL_SWAP_THURSDAY	목요일에서 금요일로 롤오버된 오버나잇 포지션에 대한 스왑 계산 비율 (SYMBOL_SWAP_LONG 또는 SYMBOL_SWAP_SHORT)	double
SYMBOL_SWAP_FRIDAY	금요일에서 토요일로 롤오버된 오버나잇 포지션에 대한 스왑 계산 비율 (SYMBOL_SWAP_LONG 또는 SYMBOL_SWAP_SHORT)	double
SYMBOL_SWAP_SATURDAY	토요일에서 일요일로 롤오버된 오버나잇 포지션에 대한 스왑 계산 비율 (SYMBOL_SWAP_LONG 또는	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	SYMBOL_SWAP_SHORT)	
SYMBOL_MARGIN_INITIAL	초기 마진은 한 로트의 부피로 한 자리를 여는 데 필요한 마진 통화의 금액을 의미합니다. 고객이 시장에 진입할 때 자산을 확인하는 데 사용합니다. SymbolInfoMarginRate() 함수는 주문 유형 및 방향에 따라 충전된 마진의 양에 대한 데이터를 제공합니다.	double
SYMBOL_MARGIN_MAINTENANCE	유지 마진. 설정된 경우 하나의 로트에서 충전된 기호의 마진 통화로 마진 금	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	액을 설정합니다. 계좌 상태가 변경될 때 고객의 자산을 확인하는 데 사용됩니다. 유지 마진이 0인 경우 초기 마진이 사용됩니다. SymbolInfoMarginRate() 함수는 주문 유형 및 방향에 따라 충전된 마진의 양에 대한 데이터를 제공합니다.	
SYMBOL_SESSION_VOLUME	현재 세션 거래의 요약 볼륨	double
SYMBOL_SESSION_TURNOVER	연재 세션의 요약 전환	double
SYMBOL_SESSION_INTEREST	열린 이자 요약	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	현재 매수 주문 볼륨	double
SYMBOL_SESSION_SELL_ORDERS_VOLUME	현재 매도 주문	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	볼륨	
SYMBOL_SESSION_OPEN	현재 세션의 시가	double
SYMBOL_SESSION_CLOSE	현재 세션의 종가	double
SYMBOL_SESSION_AW	현재 세션의 평균 가중 가격	double
SYMBOL_SESSION_PRICE_SETTLEMENT	현재 세션의 결산 가격	double
SYMBOL_SESSION_PRICE_LIMIT_MIN	현재 세션의 최소 가격	double
SYMBOL_SESSION_PRICE_LIMIT_MAX	현재 세션의 최대 가격	double
SYMBOL_MARGIN_HEDGED	위험회피 포지션의 1 로트당 계약의 크기 또는 마진 볼륨(1 개 기호의 반대 방향의 포지션). 위험회피 포지션에 대해서는 2 가지 마진 계산 방법이 가능합니다. 계산 방법은 브로커에 의	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	해 정의됩니다. 기초 계산: • 기호에 대해 초기 마진 (SYMBOL_MARGIN_INITIAL) 이 특정된 경우, 위험 회피 마진은 절대값 (화폐 용어) 으로 지정됩니다. • 초기 마진이 명시되지 않은 경우 (equal to 0), SYMBOL_MARGIN_HEDGED 는 계약의 크기와 동일하	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	며 금 융 상 품 (SYMBOL_TRADE_CALC_MODE) 의 유 형에 따라 적절 한 공 시그 로 마 진을 계산 하는 데 사 용됩 니다. 가장 큰 위치에 대한 계 산: • SYMBOL_MARGIN_HEDGED 값은 고려 되지 않습 니다 ,. • 기호 의 모 든 단 기 및 장기 포지 션의 볼륨 이 계 산됩 니다.	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	• 각 방향에 대해 가중 평가 시가와 예금 통화로의 가중 평균 환산율을 계산합니다. • 다음으로 기호 유형 (SYMBOL_TRADE_CALC_MODE)에 따라 선택한 적절한 공식을 사용하여 단기 및 장기 부분에 대해 마진이 계산됩니다. • 값 중 가장 큰 값이 마진으로 사	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	용됩니다.	
SYMBOL_PRICE_CHANGE	전 거래일 대비 현재 가격 변동률(%)	double
SYMBOL_PRICE_VOLATILITY	가격 변동성(%)	double
SYMBOL_PRICE_THEORETICAL	이론적 옵션 가격	double
SYMBOL_PRICE_DELTA	옵션/보증 델타(delta)는 기본자산 가격이 1씩 변경될 때 옵션 가격이 변경되는 값을 표시합니다.	double
SYMBOL_PRICE_THETA	옵션/보증 세타(Theta)는 일시적 해체로 인해 옵션 가격이 매일 손실되는 포인트 수를 나타냅니다(예: 만료일이 다가올 때)	double
SYMBOL_PRICE_GAMMA	옵션/보증 감마(gamma)는 델타	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	변화율(옵션 프리미엄 변경 속도)을 나타냅니다	
SYMBOL_PRICE_VEGA	옵션/보증 베가(vega)는 변동성이 1% 변경될 때 옵션 가격이 변경되는 포인트 수를 나타냅니다	double
SYMBOL_PRICE_RHO	옵션/보증 로우(rho)는 이자율의 1% 변동에 대한 이론적 옵션 가격의 민감도를 반영합니다	double
SYMBOL_PRICE_OMEGA	옵션/보증 오메가(omega). 옵션 탄력성은 옵션 가격의 상대적 백분율 변동을 기본 자산 가격의 백분	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	율로 나타냅니다	
SYMBOL_PRICE_SENSITIVITY	옵션/보증 민감도는 옵션의 기본 자산 가격이 옵션 가겨이 1 포인트 변동하도록 변경해야 하는 포인트 수로 표시합니다.	double

다음 함수의 경우: SymbolInfoString()

ENUM_SYMBOL_INFO_STRING

식별자	설명	유형
SYMBOL_BASIS	파생상품의 기본 자산	string
SYMBOL_CATEGORY	재무 기호가 속하는 섹터 또는 카테고리의 이름	string
SYMBOL_COUNTRY	재무 기호가 속한 국가	string
SYMBOL_SECTOR_NAME	재무 기호가 속한 경제 부문	string
SYMBOL_INDUSTRY_NAME	재무 기호가 속한 산업 또는 산업 분야	string
SYMBOL_CURRENCY_BASE	기호의 기본 통화	string

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
SYMBOL_CURRENCY_PROFIT	이익 통화	string
SYMBOL_CURRENCY_MARGIN	마진 통화	string
SYMBOL_BANK	현재 시세의 피더	string
SYMBOL_DESCRIPTION	기호 설명	string
SYMBOL_EXCHANGE	재무 기호가 거래되는 거래소의 이름	string
SYMBOL_FORMULA	사용자 지정 기호 가격에 사용되는 공식 공식에 사용된 재무 기호의 이름이 숫자로 시작되거나 특수 문자 (">", ".", "-", "&", "#" 등)를 포함할 경우 이 기호 이름 주위에 따옴표를 사용해야 합니다. • 합성 기호: "@ESU19"/EUR	string

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	CAD 캘린더스프레드: "Si-9.13" - "Si-6.13" 유로지수: 34.388057266*po	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	w(EURUSD,0.31555)*pow(EURGBP,0.30556)*pow(EURJPY,0.	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	1891)*pow(EURCHF,0.11113)*pow(EURSEK,0.07785)	
SYMBOL_ISIN	ISIN 시스템의 기호 이름(국제 증권 식별 기호). 국	string

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	제 증권 식별 번호는 증권을 고유하게 식별하는 12 자리 영숫자 코드입니다. 이 기호 속성의 존재는 거래 서버의 측면에서 결정됩니다,	
SYMBOL_PAGE	기호 정보가 들어 있는 웹 페이지의 주소입니다. 이 주소는 터미널에서 기호 속성을 볼 때 링크로 표시됩니다.	string
SYMBOL_PATH	기호 트리의 경로	string

기호 가격 차트는 입찰 가격 또는 최종 가격을 기준으로 할 수 있습니다. 기호 차트에 대해 선택한 가격은 터미널의 막대 생성 및 표시에도 영향을 미칩니다. SYMBOL_CHART_MODE 속성의 가능한 값은 ENUM_SYMBOL_CHART_MODE에 설명되어 있습니다

ENUM_SYMBOL_CHART_MODE

식별자	설명
SYMBOL_CHART_MODE_BID	막대는 입찰 가격을 기준으로 합니다
SYMBOL_CHART_MODE_LAST	막대는 최종 가격을 기준으로 합니다

각 기호에 대해 보류 중인 주문의 여러 만료 모드를 지정할 수 있습니다. 플래그가 각 모드에 일치합니다. 로직 **OR** (|) 연산을 사용하여 플래그를 결합할 수 있습니다(예: SYMBOL_EXPIRATION_GTC| SYMBOL_EXPIRATION_SPECIFIED). 기호에 특정 모드가 허용되는지 여부를 확인하려면 로직 **AND** (&) 의 결과를 모드 플래그와 비교해야 합니다.

© 2000-2025, MetaQuotes Ltd.

기호에 대해 SYMBOL_EXPIRATION_SPECIFIED 플래그가 지정된 경우 보류 중인 주문을 보내는 동안 이 보류 중인 주문이 유효한 시점을 지정할 수 있습니다.

식별자	값	설명
SYMBOL_EXPIRATION_GTC	1	주문이 명시적으로 취소될 때까지 무제한 기간 동안 유효합니다
SYMBOL_EXPIRATION_DAY	2	주문은 오늘까지 유효합니다
SYMBOL_EXPIRATION_SPECIFIED	4	만료 시간은 주문에 표시되어 있습니다
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	만료 날짜는 주문에 지정되어 있습니다

예:

```
//+------------------------------------------------------------------+
//| 지정된 만료 모드가 허용되는지 확인합니다                                  |
//+------------------------------------------------------------------+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
  {
//--- 허용된 만료 모드를 설명하는 속성 값 가져오기
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- exp_type 모드가 허용되는 경우 true를 반환
   return((expiration&exp_type)==exp_type);
  }
```

SYMBOL_EXPIRATION_MODE 속성이 SYMBOL_EXPIRATION_GTC (취소될 때까지 양호)로 설정된 경우 보류 중인 주문의 만료는 물론 손실 중지/이익 취득 주문도 ENUM_SYMBOL_ORDER_GTC_MODE 열거를 사용하여 추가로 설정해야 합니다.

ENUM_SYMBOL_ORDER_GTC_MODE

식별자	설명
SYMBOL_ORDERS_GTC	보류 중인 주문 및 손실 중지/이익 취득 수준은 명시적 취소 전까지 무제한으로 유효합니다
SYMBOL_ORDERS_DAILY	주문은 1 거래일 동안 유효합니다. 결국 모든 손실 중지 및 이익 취득 수준과 보류 중인 주문이 삭제됩니다.
SYMBOL_ORDERS_DAILY_EXCLUDING_STOPS	거래일이 변경되면 보류 중인 주문만 삭제되고 손실 중지 및 이익 취득 수준은 유지됩니다.

주문을 전송할 때 주문에 설정된 볼륨에 대한 작성 방침을 지정할 수 있습니다. 표에 각 기호에 대해 허용되는 주문 작성 모드가 지정되어 있습니다. 플래그를 결합하여 하나의 기호에 대해 여러 모드를 설정할 수 있습니다. 플래그는 로직 **OR** (|)의 작동으로 결합할 수 있습니다(예:SYMBOL_FILLING_FOK|

© 2000-2025, MetaQuotes Ltd.

SYMBOL_FILLING_IOC). 기호에 특정 모드가 허용되는지 여부를 확인하려면 로직 **AND (&)** 의 결과를 모드 플래그와 비교해야 합니다.

채우기 방침	식별자	값	설명
FOK(Fill Or Kill)	SYMBOL_FILLING_FOK	1	이 방침은 지정된 볼륨에서만 거래를 실행할 수 있음을 의미합니다. 현재 시장에서 금융 상품의 필요 금액이 없는 경우 주문이 실행되지 않습니다. 현재 시장에서 제공되는 몇 가지 제안을 사용하여 필요한 수량을 채울 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

채우기 방침	식별자	값	설명
IOC(Immediate Or Cancel)	SYMBOL_FILLING_IOC	2	이 경우 트레이더는 주문서에 명시된 범위 내에서 시장에서 최대 가능 물량으로 거래를 실행하기로 합의합니다. 주문이 완전히 채워지지 않을 경우, 주문 가능 수량이 채워지고 나머지 수량은 취소됩니다. IOC의 명령 사용 가능성은 거래 서버에서

© 2000-2025, MetaQuotes Ltd.

채우기 방침	식별자	값	설명
			결정됩니다.
Passive	SYMBOL_FILLING_BOC	4	BOC(Book-or-Cancel) 정책은 주문이 시장 심도에서만 접수 가능하고 즉시 실행될 수 없다고 가정합니다. 주문이 주문을 한 즉시 실행될 수 있는 경우 주문은 취소됩니다. 실제로 이 실행 정책은 접수된 주문의 가격이 현재

© 2000-2025, MetaQuotes Ltd.

채우기 방침	식별자	값	설명
			시장보다 나쁠 때만 지정될 수 있습니다. BoC 주문은 패시브 트레이딩을 구현하는 데 사용되므로 주문이 즉시 실행되지 않고 현재 유동성에 영향을 미치지 않습니다. 리밋 및 스탑리밋 주문만 지원됩니다 .SYMBOL_ORDER_MODE 플래그

© 2000-2025, MetaQuotes Ltd.

채우기 방침	식별자	값	설명
			는 SYMBOL_ORDER_LIMIT 및/또는 SYMBOL_ORDER_STOP_LIMIT 값을 포함해야 합니다.
반환	식별자 없음		이 방침은 시장 주문(매수 및 매도), 제한 및 제한 중지 주문, 시장 또는 교환 실행이 포함된 기호에만 사용됩니다. 시장 일부 충당이나 잔여 물량으로 한정

© 2000-2025, MetaQuotes Ltd.

채우기 방침	식별자	값	설명
			주문은 취소되지 않고 추가 처리됩니다.

***요청 및 인스턴트 실행 모드에서는 FOK 방침이 시장 주문에서 항상 사용되며, 반환 정책은 제한 주문에 항상 사용됩니다. 이 경우, OrderSend 또는 OrderSendAsync를 사용하여 주문을 전송할 때 주문 채우기 방침을 지정할 필요가 없습니다.

마켓 및 거래소 실행 모드에서는 모든 주문 유형에 대해 반환 정책이 항상 허용됩니다. 다른 정책의 허용 여부를 확인하려면 SYMBOL_FILLING_FOK 및 SYMBOL_FILLING_IOC 속성을 사용합니다.

예:

```
//+------------------------------------------------------------------+
//| 지정된 채우기 모드가 허용되는지 확인                                |
//+------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
//--- 허용된 채우기 모드를 설명하는 속성 값 가져오기
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- fill_type 모드가 허용되는 경우 true를 반환
   return((filling & fill_type)==fill_type);
  }
```

거래 요청을 OrderSend() 함수를 사용하여 전송하는 경우 일부 작업에 대해 ENUM_ORDER_TYPE 열거의 주문 유형을 지정해야 합니다. 특정 기호에는 일부 유형의 주문이 허용되지 않을 수 있습니다. SYMBOL_ORDER_MODE 속성은 허용되는 순서 유형의 플래그를 설명합니다.

식별자	값	설명
SYMBOL_ORDER_MARKET	1	시장 주문 허용(매수 및 매도)
SYMBOL_ORDER_LIMIT	2	주문 제한 허용(매수 제한 및 매도 제한)
SYMBOL_ORDER_STOP	4	중지 주문이 허용(매수 중지 및 매도 중지)
SYMBOL_ORDER_STOP_LIMIT	8	중지 제한 주문이 허용(매수 중지 제한 및 매도 중지 제한)
SYMBOL_ORDER_SL	16	손실 중지 허용됨
SYMBOL_ORDER_TP	32	이익 취득 허용됨

© 2000-2025, MetaQuotes Ltd.

식별자	값	설명
SYMBOL_ORDER_CLOSEBY	64	Close By 작동은 허용됩니다. 즉, 같은 금융 상품에서 반대 방향으로 열린 다른 포지션에 의해 포지션을 닫을 수 있습니다. 자산 위험회피 회계 시스템으로 계정에 대해 설정됩니다 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)

예:

```
//+------------------------------------------------------------------+
//| 함수는 기호에 허용된 주문 유형을 인쇄합니다                                  |
//+------------------------------------------------------------------+
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- 허용된 주문 유형을 설명하는 속성 값 수신
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- 시장 주문 확인 (마켓 실행)
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": 시장 주문 허용(매수 및 매도)");
//--- 제한 주문 확인
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": 매수 제한 및 매도 제한이 허용됨");
//--- 중지 주문 확인
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": 매수 중지 및 매도 중지가 허용됨");
//--- 중지 제한 주문 확인
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": 매수 중지 제한 및 매도 중지 제한 허용됨");
//--- 손실 중지 주문이 허용되는지 확인
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
      Print(symbol+": 손실 중지 주문이 허용됨");
//--- 이익 취득 주문이 허용되는지 확인
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
      Print(symbol+": 이익 취득 주문이 허용됨");
//--- 반대편의 포지션을 닫을 수 있는지 확인
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_CLOSEBY)
      Print(symbol+": 허용 기준 닫기");
//---
  }
```

ENUM_SYMBOL_CALC_MODE 열거는 기호의 마진 요구 사항 계산 방법에 대한 정보를 얻는 데 사용됩니다.

ENUM_SYMBOL_CALC_MODE

© 2000-2025, MetaQuotes Ltd.

식별자	설명	공식
SYMBOL_CALC_MODE_FOREX	외환 모드 – 외환의 이익 및 마진 계산	마진: Lots * Contract_Size / Leverage * Margin_Rate 이익: (close_price – open_price) * Contract_Size*Lots
SYMBOL_CALC_MODE_FOREX_NO_LEVERAGE	외환 무 레버리지 모드 – 외환 기호에 대한 수익 및 마진 계산 (레버리지를 고려하지 않음)	마진: Lots * Contract_Size * Margin_Rate 이익: (close_price – open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_FUTURES	선물 모드 – 선물의 마진 및 이익 계산	마진: Lots * InitialMargin * Margin_Rate 이익: (close_price – open_price) * TickPrice / TickSize*Lots
SYMBOL_CALC_MODE_CFD	CFD 모드 - CFD 의 마진 및 이익 계산	마진: Lots * ContractSize * MarketPrice * Margin_Rate 이익: (close_price – open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_CFDINDEX	CFD 지수 모드 – 지수별	마진: (Lots * ContractSize * MarketPrice) * TickPrice / TickSize * Margin_Rate 이익: (close_price – open_price) * Contract_Size * Lots

© 2000–2025, MetaQuotes Ltd.

식별자	설명	공식
	CFD 의 마진 및 이익	
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD 레버리지 모드 – 레버리지 트레이딩에서 CFD 의 마진 및 이익 계산	마진: (Lots * ContractSize * MarketPrice) / Leverage * Margin_Rate 이익: (close_price-open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_STOCKS	교환 모드 – 증권 거래의 이윤 및 이익 계산	마진: Lots * ContractSize * LastPrice * Margin_Rate 이익: (close_price – open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_FUTURES	선물 모드 – 증권 거래소에서 선물 계약을 거래할 때의 마진 및 이익 계산	마진: Lots * InitialMargin * Margin_Rate or Lots * MaintenanceMargin * Margin_Rate 이익: (close_price – open_price) * Lots * TickPrice / TickSize
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS 선물 모드	마진: Lots * InitialMargin * Margin_Rate or Lots * MaintenanceMargin * Margin_Rate

© 2000-2025, MetaQuotes Ltd.

식별자	설명	공식
	- calculation of margin and profit for trading futures contracts on FORTS에 대한 선물 계약 거래의 마진 및 이익 계산. 다음 규칙에 따라 마진 할인 편차만큼 마진을 줄일 수 있습니다: 1. 장기 포지션(매수 주문)의 가격이 예상 가격보다	이익: (close_price - open_price) * Lots * TickPrice / TickSize

© 2000-2025, MetaQuotes Ltd.

식별자	설명	공식
	낮은 경우, MarginDiscount = Lots*((PriceSettle-PriceOrder) *TickPrice/TickSize) 2. 단기 포지션(매도 주문)의 가격 예상 가격을 초과할 경우, MarginDiscount = Lots*((PriceOrder-PriceSettle) *TickPrice/TickSize) 위치: ○ Pr	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	공식
	iceSettle - 이전 세션의 추정 (정리) 가격; ○ PriceOrder - 주문에 설정된 평균 가중 포지션 가	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	공식
	걱또는공개가격(요청);○TickPrice-틱가격(1포인트별가격변경비용)○TickSize-틱크	

© 2000-2025, MetaQuotes Ltd.

식별자	설명	공식
	가 (최소가격변경단계)	
SYMBOL_CALC_MODE_EXCH_BONDS	교환 채권 모드 – 증권거래소에서 채권을 거래하기 위한 마진 및 이익 계산	마진: Lots * ContractSize * FaceValue * open_price * /100 이익: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize
SYMBOL_CALC_MODE_EXCH_STOCKS_MOEX	교환 MOEX 증권 모드 – MOEX의 증권 거래 마진 및 이익 계산	마진: Lots * ContractSize * LastPrice * Margin_Rate 이익: (close_price – open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_BONDS_MOEX	교환 MOEX 채권 모드 – MOEX의 채권	마진: Lots * ContractSize * FaceValue * open_price * /100 이익: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

© 2000–2025, MetaQuotes Ltd.

식별자	설명	공식
	거래의 이익과 마진 계산	
SYMBOL_CALC_MODE_SERV_COLLATERAL	담보 모드 – 기호는 거래 계좌에서 거래할 수 없는 자산으로 사용됩니다. 오픈 포지션의 시장 가치는 물량, 현재 시세, 계약 규모, 유동성 비율을 기준으로 산출됩니다. 이 가치는 자산에 포함되며, 자산은 자본에 추가	마진: no 이익: no 시장 가치: Lots*ContractSize*MarketPrice*LiqudityRate

© 2000-2025, MetaQuotes Ltd.

식별자	설명	공식
	됩니다. 이러한 기호의 오픈 포지션은 여유 마진 금액을 증가시키며, 거래 가능한 금융상품의 오픈 포지션에 대한 추가 마진(담보)으로 사용됩니다.	

여러가지 기호 거래 모드가 있습니다. 특정 기호의 거래 모드에 대한 정보는 ENUM_SYMBOL_TRADE_MODE 열거 값에 반영됩니다.

ENUM_SYMBOL_TRADE_MODE

식별자	설명
SYMBOL_TRADE_MODE_DISABLED	기호에 대해 거래가 비활성화되었습니다
SYMBOL_TRADE_MODE_LONGONLY	장기 포지션만 허용됨
SYMBOL_TRADE_MODE_SHORTONLY	단기 포지션만 허용됨
SYMBOL_TRADE_MODE_CLOSEONLY	포지션 닫기 작업만 허용됨
SYMBOL_TRADE_MODE_FULL	거래 제한 없음

© 2000-2025, MetaQuotes Ltd.

특정 기호에 대해 가능한 거래 실행 모드는 ENUM_SYMBOL_TRADE_EXECUTION 열거에 정의되어 있습니다.

ENUM_SYMBOL_TRADE_EXECUTION

식별자	설명
SYMBOL_TRADE_EXECUTION_REQUEST	요청별 실행
SYMBOL_TRADE_EXECUTION_INSTANT	인스턴트 실행
SYMBOL_TRADE_EXECUTION_MARKET	마켓 실행
SYMBOL_TRADE_EXECUTION_EXCHANGE	거래소 실행

포지션 전송 시 스왑 계산 방법은 ENUM_SYMBOL_SWAP_MODE 열거에 지정되어 있습니다. 스왑 계산 방법은 SYMBOL_SWAP_LONG 및 SYMBOL_SWAP_SHORT 매개변수 측정 단위를 결정합니다. 예를 들어 스왑이 클라이언트 예금 통화로 청구되는 경우 이러한 매개변수의 값은 클라이언트 예금 통화의 금액으로 지정됩니다.

ENUM_SYMBOL_SWAP_MODE

식별자	설명
SYMBOL_SWAP_MODE_DISABLED	스왑 사용 안 함(스왑 없음)
SYMBOL_SWAP_MODE_POINTS	스왑이 포인트로 청구됩니다
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	스왑은 기호의 기준 통화로 화폐로 청구됩니다
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	스왑은 기호의 마진 통화로 화폐로 청구됩니다
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	스왑은 화폐, 클라이언트 예금 통화로 청구됩니다
SYMBOL_SWAP_MODE_CURRENCY_PROFIT	스왑은 수익 계산 통화로 청구됩니다
SYMBOL_SWAP_MODE_INTEREST_CURRENT	스왑은 스왑 계산 시 금융상품 가격에서 지정된 연간 이자로 청구됩니다(표준 은행 연도는 360일)
SYMBOL_SWAP_MODE_INTEREST_OPEN	스왑은 포지션 공개 가격에서 지정된 연간 이자로 청구됩니다(표준 은행 연도는 360일)
SYMBOL_SWAP_MODE_REOPEN_CURRENT	스왑은 포지션을 다시 열어서 청구됩니다. 거래일이 끝나면 해당 포지션은 닫힙니다. 다음 날 마감 가격 +/- 지정된 포인트 수만큼 다시 열립니다 (매개변수 SYMBOL_SWAP_LONG 및 SYMBOL_SWAP_SHORT)
SYMBOL_SWAP_MODE_REOPEN_BID	스왑은 포지션을 다시 열어서 청구됩니다. 거래일이 끝나면 해당 포지션은 닫힙니다. 다음 날 현재 입찰 가격 +/- 지정된 포인트 수만큼 다시 열립니다 (매개변수 SYMBOL_SWAP_LONG 및 SYMBOL_SWAP_SHORT)

© 2000-2025, MetaQuotes Ltd.

ENUM_DAY_OF_WEEK 열거 값은 요일을 지정하는 데 사용됩니다.

ENUM_DAY_OF_WEEK

식별자	설명
SUNDAY	일요일
MONDAY	월요일
TUESDAY	화요일
WEDNESDAY	수요일
THURSDAY	목요일
FRIDAY	금요일
SATURDAY	토요일

옵션은 특정일 또는 그 이전에 특정가격으로 기초자산(재화, 주식, 선물 등)을 매수하거나 매도할 권리를 주지만 의무가 아닌 계약입니다. 다음 열거에서는 옵션 유형 및 옵션에서 발생하는 권리를 포함하여 옵션 속성을 설명합니다.

ENUM_SYMBOL_OPTION_RIGHT

식별자	설명
SYMBOL_OPTION_RIGHT_CALL	콜 옵션은 특정 가격으로 자산을 구입할 수 있는 권리를 부여합니다
SYMBOL_OPTION_RIGHT_PUT	풋 옵션은 특정 가격으로 자산을 판매할 수 있는 권리를 부여합니다

ENUM_SYMBOL_OPTION_MODE

식별자	설명
SYMBOL_OPTION_MODE_EUROPEAN	유러피언 옵셔은 지정된 날짜(만료, 실행일, 전달일)에만 행사할 수 있습니다
SYMBOL_OPTION_MODE_AMERICAN	아메리칸 옵션은 모든 거래일 또는 만료 전에 행사할 수 있습니다. 매수인이 옵션을 행사할 수 있는 기간이 지정되어 있습니다

금융상품은 경제 분야별로 분류됩니다. 경제 섹터는 특정 특성, 경제 목표, 기능 및 행위를 가진 경제 활동의 일부로서, 이 섹터를 경제의 다른 파트와 분리할 수 있습니다. ENUM_SYMBOL_SECTOR에는 트레이딩 상품이 속할 수 있는 경제 섹터가 나열됩니다.

ENUM_SYMBOL_SECTOR

© 2000-2025, MetaQuotes Ltd.

ID	설명
SECTOR_UNDEFINED	정의되지 않음
SECTOR_BASIC_MATERIALS	기본 재료
SECTOR_COMMUNICATION_SERVICES	커뮤니케이션 서비스
SECTOR_CONSUMER_CYCLICAL	소비자 순환
SECTOR_CONSUMER_DEFENSIVE	소비자 방어
SECTOR_CURRENCY	통화
SECTOR_CURRENCY_CRYPTO	암호화폐
SECTOR_ENERGY	에너지
SECTOR_FINANCIAL	재정
SECTOR_HEALTHCARE	의료
SECTOR_INDUSTRIALS	산업
SECTOR_REAL_ESTATE	부동산
SECTOR_TECHNOLOGY	기술
SECTOR_UTILITIES	공익사업

각 금융상품은 특정 유형의 산업 또는 경제 지점에 할당할 수 있습니다. 산업은 원자재, 재화 또는 서비스와 밀접하게 관련된 집합을 생산하는 경제의 한 분야입니다. ENUM_SYMBOL_INDUSTRY는 트레이딩 상품이 속할 수 있는 업종을 나열합니다.

ENUM_SYMBOL_INDUSTRY

ID	설명
INDUSTRY_UNDEFINED	정의되지 않음
기본 재료	
INDUSTRY_AGRICULTURAL_INPUTS	농업 투입물
INDUSTRY_ALUMINIUM	알루미늄
INDUSTRY_BUILDING_MATERIALS	건축 자재
INDUSTRY_CHEMICALS	화학
INDUSTRY_COKING_COAL	점결탄
INDUSTRY_COPPER	구리
INDUSTRY_GOLD	금
INDUSTRY_LUMBER_WOOD	목재 및 목재 생산

© 2000-2025, MetaQuotes Ltd.

ID	설명
INDUSTRY_INDUSTRIAL_METALS	기타 산업용 금속 및 광업
INDUSTRY_PRECIOUS_METALS	기타 귀금속 및 광업
INDUSTRY_PAPER	지류 및 종이 제품
INDUSTRY_SILVER	은
INDUSTRY_SPECIALTY_CHEMICALS	특수 화학
INDUSTRY_STEEL	철강
커뮤니케이션 서비스	
INDUSTRY_ADVERTISING	광고 대행사
INDUSTRY_BROADCASTING	방송
INDUSTRY_GAMING_MULTIMEDIA	전자 게임 및 멀티미디어
INDUSTRY_ENTERTAINMENT	엔터테인먼트
INDUSTRY_INTERNET_CONTENT	인터넷 컨텐츠 및 정보
INDUSTRY_PUBLISHING	출판
INDUSTRY_TELECOM	텔레콤 서비스
소비자 순환	
INDUSTRY_APPAREL_MANUFACTURING	의류 제조
INDUSTRY_APPAREL_RETAIL	의류 소매
INDUSTRY_AUTO_MANUFACTURERS	자동차 제조
INDUSTRY_AUTO_PARTS	자동차 부품
INDUSTRY_AUTO_DEALERSHIP	자동차 및 트럭 대리점
INDUSTRY_DEPARTMENT_STORES	백화점
INDUSTRY_FOOTWEAR_ACCESSORIES	제화 및 악세서리
INDUSTRY_FURNISHINGS	가구, 고정장치 및 어플라이언스
INDUSTRY_GAMBLING	겜블링
INDUSTRY_HOME_IMPROV_RETAIL	주택 개선 소매
INDUSTRY_INTERNET_RETAIL	인터넷 소매
INDUSTRY_LEISURE	레저
INDUSTRY_LODGING	투숙
INDUSTRY_LUXURY_GOODS	럭셔리
INDUSTRY_PACKAGING_CONTAINERS	포장 및 컨테이너

© 2000-2025, MetaQuotes Ltd.

ID	설명
INDUSTRY_PERSONAL_SERVICES	개인 서비스
INDUSTRY_RECREATIONAL_VEHICLES	여가용 차량
INDUSTRY_RESIDENT_CONSTRUCTION	주거 건설
INDUSTRY_RESORTS_CASINOS	리조트 및 카지노
INDUSTRY_RESTAURANTS	레스토랑
INDUSTRY_SPECIALTY_RETAIL	전문 소매
INDUSTRY_TEXTILE_MANUFACTURING	섬유 제조
INDUSTRY_TRAVEL_SERVICES	여행 서비스
소비자 방어	
INDUSTRY_BEVERAGES_BREWERS	음료 – 양조장
INDUSTRY_BEVERAGES_NON_ALCO	음료 – 비주류
INDUSTRY_BEVERAGES_WINERIES	음료 – 와이너리 및 증류소
INDUSTRY_CONFECTIONERS	과자류
INDUSTRY_DISCOUNT_STORES	할인점
INDUSTRY_EDUCATION_TRAINIG	교육 및 훈련 서비스
INDUSTRY_FARM_PRODUCTS	농장 제품
INDUSTRY_FOOD_DISTRIBUTION	식품 유통
INDUSTRY_GROCERY_STORES	식료품점
INDUSTRY_HOUSEHOLD_PRODUCTS	가정용 및 개인용 제품
INDUSTRY_PACKAGED_FOODS	포장식품
INDUSTRY_TOBACCO	담배
에너지	
INDUSTRY_OIL_GAS_DRILLING	석유 및 가스 시추
INDUSTRY_OIL_GAS_EP	석유 및 가스 추출 및 처리
INDUSTRY_OIL_GAS_EQUIPMENT	석유 및 가스 장비 및 서비스
INDUSTRY_OIL_GAS_INTEGRATED	석유 및 가스 통합
INDUSTRY_OIL_GAS_MIDSTREAM	석유 및 가스 중류
INDUSTRY_OIL_GAS_REFINING	정유, 가스 정제 및 마케팅
INDUSTRY_THERMAL_COAL	열탄
INDUSTRY_URANIUM	우라늄

© 2000-2025, MetaQuotes Ltd.

ID	설명
재정	
INDUSTRY_EXCHANGE_TRADED_FUND	거래소 거래펀드
INDUSTRY_ASSETS_MANAGEMENT	자산 관리
INDUSTRY_BANKS_DIVERSIFIED	은행 – 종합
INDUSTRY_BANKS_REGIONAL	은행 – 지역
INDUSTRY_CAPITAL_MARKETS	자본 시장
INDUSTRY_CLOSE_END_FUND_DEBT	마감 펀드 – 부채
INDUSTRY_CLOSE_END_FUND_EQUITY	마감 펀드 – 지분
INDUSTRY_CLOSE_END_FUND_FOREIGN	마감 펀드 – 외환
INDUSTRY_CREDIT_SERVICES	신용 서비스
INDUSTRY_FINANCIAL_CONGLOMERATE	금융 대기업
INDUSTRY_FINANCIAL_DATA_EXCHANGE	재무 데이터 및 증권 거래소
INDUSTRY_INSURANCE_BROKERS	보험중개사
INDUSTRY_INSURANCE_DIVERSIFIED	보험 – 종합
INDUSTRY_INSURANCE_LIFE	보험 – 생명
INDUSTRY_INSURANCE_PROPERTY	보험 – 재산 및 상해
INDUSTRY_INSURANCE_REINSURANCE	보험 – 재보험
INDUSTRY_INSURANCE_SPECIALTY	보험 – 전문분야
INDUSTRY_MORTGAGE_FINANCE	주택담보대출금융
INDUSTRY_SHELL_COMPANIES	셸 회사
의료	
INDUSTRY_BIOTECHNOLOGY	바이오테크놀로지
INDUSTRY_DIAGNOSTICS_RESEARCH	진단 및 연구
INDUSTRY_DRUGS_MANUFACTURERS	제약 회사 – 일반
INDUSTRY_DRUGS_MANUFACTURERS_SPEC	제약 회사 – 전문 및 일반
INDUSTRY_HEALTHCARE_PLANS	의료 계획
INDUSTRY_HEALTH_INFORMATION	건강 정보 서비스
INDUSTRY_MEDICAL_FACILITIES	의료 시설
INDUSTRY_MEDICAL_DEVICES	의료 기기
INDUSTRY_MEDICAL_DISTRIBUTION	의료 유통

© 2000-2025, MetaQuotes Ltd.

ID	설명
INDUSTRY_MEDICAL_INSTRUMENTS	의료 기기 및 소모품
INDUSTRY_PHARM_RETAILERS	Pharmaceutical retailers
산업	
INDUSTRY_AEROSPACE_DEFENSE	항공우주 및 방위
INDUSTRY_AIRLINES	항공
INDUSTRY_AIRPORTS_SERVICES	공항 및 공항 서비스
INDUSTRY_BUILDING_PRODUCTS	빌딩 제품 및 장비
INDUSTRY_BUSINESS_EQUIPMENT	비ㅈ니스 장비 및 소모품
INDUSTRY_CONGLOMERATES	대기업
INDUSTRY_CONSULTING_SERVICES	컨설팅 서비스
INDUSTRY_ELECTRICAL_EQUIPMENT	전자 기기 및 부품
INDUSTRY_ENGINEERING_CONSTRUCTION	엔지니어링 및 건설
INDUSTRY_FARM_HEAVY_MACHINERY	농중공업 기계
INDUSTRY_INDUSTRIAL_DISTRIBUTION	산업 유통
INDUSTRY_INFRASTRUCTURE_OPERATIONS	인프라 운영
INDUSTRY_FREIGHT_LOGISTICS	화물 및 물류 통합
INDUSTRY_MARINE_SHIPPING	해상 운송
INDUSTRY_METAL_FABRICATION	금속 제조
INDUSTRY_POLLUTION_CONTROL	오염 및 처리 관리
INDUSTRY_RAILROADS	철도
INDUSTRY_RENTAL_LEASING	임대 및 리스 서비스
INDUSTRY_SECURITY_PROTECTION	보안 및 보호 서비스
INDUSTRY_SPEALITY_BUSINESS_SERVICES	전문 사업 서비스
INDUSTRY_SPEALITY_MACHINERY	전문 산업 기계
INDUSTRY_STUFFING_EMPLOYMENT	파견 및 고용 서비스
INDUSTRY_TOOLS_ACCESSORIES	도구 및 액세서리
INDUSTRY_TRUCKING	운송
INDUSTRY_WASTE_MANAGEMENT	폐기물 관리
부동산	
INDUSTRY_REAL_ESTATE_DEVELOPMENT	부동산 – 개발

© 2000–2025, MetaQuotes Ltd.

ID	설명
INDUSTRY_REAL_ESTATE_DIVERSIFIED	부동산 – 종합
INDUSTRY_REAL_ESTATE_SERVICES	부동산 서비스
INDUSTRY_REIT_DIVERSIFIED	부동산 투자 신탁 – 종합
INDUSTRY_REIT_HEALTCARE	부동산 투자 신탁 – 의료 시설
INDUSTRY_REIT_HOTEL_MOTEL	부동산 투자 신탁 – 호텔 및 모텔
INDUSTRY_REIT_INDUSTRIAL	부동산 투산 신탁 – 산업
INDUSTRY_REIT_MORTAGE	부동산 투자 신탁 – 주택담보대출
INDUSTRY_REIT_OFFICE	부동산 투자 신탁 – 업무
INDUSTRY_REIT_RESIDENTAL	부동산 투자 신탁 – 주거
INDUSTRY_REIT_RETAIL	부동산 투자 신탁 – 소매
INDUSTRY_REIT_SPECIALITY	부동산 투자 신탁 – 전문분야
기술	
INDUSTRY_COMMUNICATION_EQUIPMENT	커뮤니케이션 장비
INDUSTRY_COMPUTER_HARDWARE	컴퓨터 하드웨어
INDUSTRY_CONSUMER_ELECTRONICS	가전제품
INDUSTRY_ELECTRONIC_COMPONENTS	전자 부품
INDUSTRY_ELECTRONIC_DISTRIBUTION	가전 및 컴퓨터 유통
INDUSTRY_IT_SERVICES	정보 기술 서비스
INDUSTRY_SCIENTIFIC_INSTRUMENTS	과학 및 기술 장비
INDUSTRY_SEMICONDUCTOR_EQUIPMENT	반도체 장비 및 재료
INDUSTRY_SEMICONDUCTORS	반도체
INDUSTRY_SOFTWARE_APPLICATION	소프트웨어 – 애플리케이션
INDUSTRY_SOFTWARE_INFRASTRUCTURE	소프트웨어 – 인프라
INDUSTRY_SOLAR	태양열
공익사업	
INDUSTRY_UTILITIES_DIVERSIFIED	공익사업 -종합
INDUSTRY_UTILITIES_POWERPRODUCERS	공익사업 – 독립 전력 생산
INDUSTRY_UTILITIES_RENEWABLE	공익사업 – 재생가능
INDUSTRY_UTILITIES_REGULATED_ELECTRIC	공익사업 – 전기 조절
INDUSTRY_UTILITIES_REGULATED_GAS	공익사업 – 가스 조절

© 2000-2025, MetaQuotes Ltd.

ID	설명
INDUSTRY_UTILITIES_REGULATED_WATER	공익사업 – 수도 조절
INDUSTRY_UTILITIES_FIRST	공익사업 서비스는 유형 열거의 시작입니다. INDUSTRY_UTILITIES_DIVERSIFIED에 해당.
INDUSTRY_UTILITIES_LAST	공익사업 서비스 유형 열거의 끝입니다. INDUSTRY_UTILITIES_REGULATED_WATER에 해당.

© 2000-2025, MetaQuotes Ltd.

계정 속성

현재 계정에 대한 정보를 얻기 위한 몇 가지 기능이 있습니다: AccountInfoInteger(), AccountInfoDouble() 및 AccountInfoString(). 함수 매개변수 값은 해당 ENUM_ACCOUNT_INFO 열거값의 값을 수락할 수 있습니다.

다음 함수의 경우: AccountInfoInteger()

ENUM_ACCOUNT_INFO_INTEGER

식별자	설명	유형
ACCOUNT_LOGIN	계좌 번호	long
ACCOUNT_TRADE_MODE	계좌 거래 모드	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	계좌 레버리지	long
ACCOUNT_LIMIT_ORDERS	허용된 최대 활성 보류 주문 수	int
ACCOUNT_MARGIN_SO_MODE	최소 허용 마진 설정 모드	ENUM_ACCOUNT_STOPOUT_MODE
ACCOUNT_TRADE_ALLOWED	다음의 거래 허용: 현재 계좌	bool
ACCOUNT_TRADE_EXPERT	다음의 거래 허용: 엑스퍼트 어드바이저	bool
ACCOUNT_MARGIN_MODE	마진 계산 모드	ENUM_ACCOUNT_MARGIN_MODE
ACCOUNT_CURRENCY_DIGITS	거래 결과를 정확하게 표시하는 데 필요한 계정 통화의 소	int

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	수 자릿수	
ACCOUNT_FIFO_CLOSE	FIFO 규칙에 의해서만 포지션을 닫을 수 있음을 나타내는 지표. 속성 값을 true로 설정하면 각 기호 포지션이 가장 오래된 기호부터 열리는 동일한 순서로 닫힙니다. 다른 순서로 포지션을 마감하려 할 경우 트레이더가 적절한 오류를 받게 됩니다. 비회피 포지션 계좌 모드 (ACCOUNT_MARGIN_MODE!=ACCOUNT_MARGIN_MODE_RE	bool

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	TAIL_HEDGING)인 경우, 속성 값은 언제나 false입니다.	

다음 함수의 경우: AccountInfoDouble()

ENUM_ACCOUNT_INFO_DOUBLE

식별자	설명	유형
ACCOUNT_BALANCE	예금 통화 계좌 잔액	double
ACCOUNT_CREDIT	예금 통화 계좌 신용	double
ACCOUNT_PROFIT	예금 통화 계좌의 현재 이익	double
ACCOUNT_EQUITY	예금 통화 계좌 자본	double
ACCOUNT_MARGIN	예금 통화에 사용된 계좌 마진	double
ACCOUNT_MARGIN_FREE	예금 통화 계좌의 여유 마진	double
ACCOUNT_MARGIN_LEVEL	계좌 마진 수준(%)	double
ACCOUNT_MARGIN_SO_CALL	마진 호출 수준. 설정된 ACCOUNT_MARGIN_S	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	O_MODE에 따라 백분율 또는 예금 통화로 표시됩니다	
ACCOUNT_MARGIN_SO_SO	마진 중지 출력 수준. 설정된 ACCOUNT_MARGIN_SO_MODE에 따라 백분율 또는 예금 통화로 표시됩니다	double
ACCOUNT_MARGIN_INITIAL	초기 마진. 보류 중인 모든 주문의 마진을 충당하기 위한 계정의 예약 금액	double
ACCOUNT_MARGIN_MAINTENANCE	유지 마진. 모든 오픈 포지션의 최소 금액을 충당하기 위한 계좌 적립 최소 자본	double
ACCOUNT_ASSETS	계좌의 현재 자산	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
ACCOUNT_LIABILITIES	계좌의 현재 유동 부채	double
ACCOUNT_COMMISSION_BLOCKED	계정에 현재 차단된 커미션 금액	double

다음 함수의 경우: AccountInfoString()

ENUM_ACCOUNT_INFO_STRING

식별자	설명	유형
ACCOUNT_NAME	클라이언트 이름	string
ACCOUNT_SERVER	거래 서버 이름	string
ACCOUNT_CURRENCY	계좌 통화	string
ACCOUNT_COMPANY	계정을 서비스하는 회사 이름	string

거래 서버에서 개설할 수 있는 계좌 유형은 여러 가지가 있습니다. MQL5 프로그램이 실행 중인 계정 유형은 ENUM_ACCOUNT_TRADE_MODE 열거를 사용하여 확인할 수 있습니다.

ENUM_ACCOUNT_TRADE_MODE

식별자	설명
ACCOUNT_TRADE_MODE_DEMO	데모 계좌
ACCOUNT_TRADE_MODE_CONTEST	콘테스트 계좌
ACCOUNT_TRADE_MODE_REAL	실제 계좌

형평성이 개방 적인 위치를 유지하는 데 충분하지 않은 경우, 스톱 아웃 상황, 즉 강제 종료가 발생합니다. 스톱 아웃이 발생하는 최소 마진 수준은 백분율 또는 통화 단위로 설정할 수 있습니다. 계정에 대해 설정된 모드를 확인하려면 ENUM_ACCOUNT_STOPOUT_MODE 열거를 사용해야 합니다.

ENUM_ACCOUNT_STOPOUT_MODE

식별자	설명
ACCOUNT_STOPOUT_MODE_PERCENT	계좌 중지 모드(%)

© 2000-2025, MetaQuotes Ltd.

식별자	설명
ACCOUNT_STOPOUT_MODE_MONEY	계좌 중지 모드(금액)

ENUM_ACCOUNT_MARGIN_MODE

식별자	설명
ACCOUNT_MARGIN_MODE_RETAIL_NETTING	OTC 시장이 "네팅" 모드의 위치를 해석하는 데 사용됩니다 (기호 하나에 대해 하나의 위치만 존재할 수 있습니다). 기호 유형에 따라 마진이 계산됩니다 (SYMBOL_TRADE_CALC_MODE).
ACCOUNT_MARGIN_MODE_EXCHANGE	교환 시장에 사용됩니다. 기호 설정에 지정된 할인에 따라 마진이 계산됩니다. 할인은 브로커가 설정하지만 거래소에서 설정한 값보다 작지는 않습니다.
ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	개별 포지션이 가능한 거래소 시장에 사용됩니다(헤지, 하나의 기호에 대해 여러 포지션이 존재할 수 있음.) 이 마진은 기호 유형 (SYMBOL_TRADE_CALC_MODE)에 기초하여 계산하며, 위험 회피 마진 (SYMBOL_MARGIN_HEDGED)을 고려합니다.

간단한 계정 정보를 출력하는 스크립트의 예입니다.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 회사명
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- 클라이언트 이름
   string name=AccountInfoString(ACCOUNT_NAME);
//--- 계좌 번호
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- 서버 이름
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- 계좌 통화
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- 데모, 콘테스트 또는 실제 계좌
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- 이제 열거의 값을 이해할 수 있는 형태로 변환합니다
   string trade_mode;
   switch(account_type)
     {
      case  ACCOUNT_TRADE_MODE_DEMO:
         trade_mode="demo";
         break;
      case  ACCOUNT_TRADE_MODE_CONTEST:
```

© 2000-2025, MetaQuotes Ltd.

```
         trade_mode="contest";
         break;
      default:
         trade_mode="real";
         break;
     }
//--- 스톱 아웃이 설정됩니다(% 또는 금액)
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteg
//--- 마진 호출 및 스톱 아웃 발생 시 수준 값을 가져오기
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- 간단한 계좌 정보 표시
   PrintFormat("클라이언트 '%s' #%d %s 계정이 '%s' 서버의 '%s'에 열려 있습니다",
               name,login,trade_mode,company,server);
   PrintFormat("Account currency - %s, MarginCall and StopOut levels are set in %s",
               currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"percentage":" m
   PrintFormat("MarginCall=%G, StopOut=%G",margin_call,stop_out);
  }
```

© 2000-2025, MetaQuotes Ltd.

통계 테스트

테스트가 끝나면 거래 결과 통계의 서로 다른 매개변수가 계산됩니다. 매개변수의 값은 ENUM_STATISTICS 열거에서 매개변수 ID를 지정하여 TesterStatistics() 함수를 사용하여 얻을 수 있습니다.

통계량을 계산하는 데 두 가지 유형의 매개변수(int 및 double)가 사용되지만 함수는 모든 값을 double 형식으로 반환합니다. double 유형의 모든 통계 값은 달리 명시되지 않는 한 기본적으로 예금 통화로 표시됩니다.

ENUM_STATISTICS

ID	통계 매개변수에 대한 설명	유형
STAT_INITIAL_DEPOSIT	최초 예금가액	double
STAT_WITHDRAWAL	계좌에서 인출된 자금	double
STAT_PROFIT	검사 후 순이익, STAT_GROSS_PROFIT 및 STAT_GROSS_LOSS의 합계 (STAT_GROSS_LOSS는 항상 0보다 작거나 같음)	double
STAT_GROSS_PROFIT	총 이익, 모든 수익(양수) 거래의 합계입니다. 값이 0보다 크거나 같습니다	double
STAT_GROSS_LOSS	총 손실, 모든 마이너스 거래의 합계입니다. 값이 0보다 작거나 같습니다	double
STAT_MAX_PROFITTRADE	최대 이익 – 모든 수익성 있는 거래 중 가장 큰 값. 값이 0보다 크거나 같습니다	double
STAT_MAX_LOSSTRADE	최대 손실 – 모든 손실 거래 중 가장 낮은 값. 값이 0보다 작거나 같습니다	double
STAT_CONPROFITMAX	일련의 수익성 거래에서 최대 이익. 값이 0보다 크거나 같습니다	double
STAT_CONPROFITMAX_TRADES	STAT_CONPROFITMAX를 형성한 거래 수(수익성 거래의 최대 이익)	int

© 2000-2025, MetaQuotes Ltd.

ID	통계 매개변수에 대한 설명	유형
STAT_MAX_CONWINS	가장 긴 일련의 수익성 거래의 총 이익	double
STAT_MAX_CONPROFIT_TRADES	가장 긴 일련의 수익성 거래 수 STAT_MAX_CONWINS	int
STAT_CONLOSSMAX	일련의 손실 거래의 최대 손실. 값이 0보다 작거나 같습니다	double
STAT_CONLOSSMAX_TRADES	STAT_CONLOSSMAX를 형성한 거래 수 (일련의 손실 거래에서 최대 손실)	int
STAT_MAX_CONLOSSES	가장 긴 일련의 손실 거래의 총 손실	double
STAT_MAX_CONLOSS_TRADES	가장 긴 일련의 손실 거래의 거래 수 STAT_MAX_CONLOSSES	int
STAT_BALANCEMIN	최소 잔액 값	double
STAT_BALANCE_DD	통화 기준의 최대 잔액 감소. 트레이딩 과정에서 잔액이 수없이 줄어들 수 있습니다. 여기서 가장 큰 값을 취합니다	double
STAT_BALANCEDD_PERCENT	잔액 감소는 통화 기준으로 최대 잔액 감소 시점에서 기록된 백분율입니다 (STAT_BALANCE_DD).	double
STAT_BALANCE_DDREL_PERCENT	최대 잔액 인출 백분율입니다. 트레이딩 과정에서 잔액은 수많은 단점을 가질 수 있으며, 각 단위에 대해 상대 인출 값(%)을 계산합니다. 가장 큰 값이 반환되었습니다	double
STAT_BALANCE_DD_RELATIVE	최대 잔액 인출 시점에 백분율로 기록된 통화 기준의 잔액 인출 (STAT_BALANCE_DDREL_PERCENT).	double

© 2000-2025, MetaQuotes Ltd.

ID	통계 매개변수에 대한 설명	유형
STAT_EQUITYMIN	최소 지분 값	double
STAT_EQUITY_DD	통화 기준의 최대 자본 감소. 거래 과정에서 자본에 수많은 하락이 나타날 수 있습니다. 여기서 가장 큰 값을 취합니다	double
STAT_EQUITYDD_PERCENT	통화 기준으로 최대 자본 감소 시점에 기록된 백분율로 인출 (STAT_EQUITY_DD).	double
STAT_EQUITY_DDREL_PERCENT	최대 자본 감소율(%) 거래 과정에서 자본은 수많은 하락을 겪을 수 있으며 각 하락가치는 백분율로 계산됩니다. 가장 큰 값이 반환되었습니다	double
STAT_EQUITY_DD_RELATIVE	최대 자본 감소 시점에 기록된 통화 기준의 자본 감소 (STAT_EQUITY_DDREL_PERCENT).	double
STAT_EXPECTED_PAYOFF	기대 수당	double
STAT_PROFIT_FACTOR	수익 계수, STAT_GROSS_PROFIT/STAT_GROSS_LOSS의 비율입니다. STAT_GROSS_LOSS=0 이면 수익 계수는 DBL_MAX와 같습니다	double
STAT_RECOVERY_FACTOR	복원 계수, STAT_PROFIT/STAT_BALANCE_DD의 비율과 동일합니다	double
STAT_SHARPE_RATIO	샤프비	double
STAT_MIN_MARGINLEVEL	마진 수준의 최소값	double
STAT_CUSTOM_ONTESTER	OnTester() 함수에 의해 반환된 계산된 사용자 지정 최적화 기준 값	double
STAT_DEALS	딜 수	int

© 2000-2025, MetaQuotes Ltd.

ID	통계 매개변수에 대한 설명	유형
STAT_TRADES	거래 수	int
STAT_PROFIT_TRADES	수익성 거래	int
STAT_LOSS_TRADES	손실 거래	int
STAT_SHORT_TRADES	단기 거래	int
STAT_LONG_TRADES	장기 거래	int
STAT_PROFIT_SHORTTRADES	수익성 단기 거래	int
STAT_PROFIT_LONGTRADES	수익성 장기 거래	int
STAT_PROFITTRADES_AVGCON	일련의 수익성 거래의 평균 길이	int
STAT_LOSSTRADES_AVGCON	일련의 손실 손실 거래의 평균 길이	int
STAT_COMPLEX_CRITERION	복잡한 최적화 기준	

© 2000-2025, MetaQuotes Ltd.

거래 상수

프로그래밍 거래 전략에 사용되는 다양한 상수는 다음과 같은 그룹으로 나뉩니다:

- 내역 데이터베이스 속성 – 기호에 대한 일반 정보 수신;
- 주문 속성 – 거래 주문에 대한 정보 획득;
- 위치 속성 – 현재 포지션에 대한 정보 얻기;
- 거래 속성 – 거래 정보 얻기;
- 거래 작업 유형 – 사용 가능한 거래 작업에 대한 설명;
- 거래 트랜잭션 유형 – 가능한 거래 유형에 대한 설명;
- DOM 거래 주문 – 요청 작업 방향에 따른 주문 분리

© 2000-2025, MetaQuotes Ltd.

내역 데이터베이스 속성

시계열에 액세스할 때 SeriesInfoInteger() 함수를 사용하여 추가 기호 정보를 가져옵니다. 필수 속성의 식별자가 함수 매개변수로 전달됩니다. 식별자는 ENUM_SERIES_INFO_INTEGER 값 중 하나일 수 있습니다.

ENUM_SERIES_INFO_INTEGER

식별자	설명	유형
SERIES_BARS_COUNT	현재 모멘트의 기호 기간에 대한 막대 수	long
SERIES_FIRSTDATE	현재 모멘트의 기호 기간의 첫 번째 날짜	datetime
SERIES_LASTBAR_DATE	기호 기간의 마지막 막대 열림 시간	datetime
SERIES_SERVER_FIRSTDATE	시간 프레임에 관계없이 서버의 기호 내역에서 첫 번째 날짜	datetime
SERIES_TERMINAL_FIRSTDATE	기간과 관계없이 클라이언트 터미널의 기호 내역에서 가장 첫 번째 날짜	datetime
SERIES_SYNCHRONIZED	기호/주기 데이터 동기화 플래그(현재 모멘트	bool

© 2000-2025, MetaQuotes Ltd.

주문 속성

거래 작업을 수행하기 위한 요청은 주문으로 공식화됩니다. 각 주문에는 다양한 읽기 속성이 있습니다. 이러한 정보는 OrderGet...() 및 HistoryOrderGet...() 함수를 사용하여 얻을 수 있습니다.

다음 함수의 경우: OrderGetInteger() 및 HistoryOrderGetInteger()

ENUM_ORDER_PROPERTY_INTEGER

식별자	설명	유형
ORDER_TICKET	주문 티켓. 각 주문에 할당된 고유 번호	long
ORDER_TIME_SETUP	주문 셋업 시간	datetime
ORDER_TYPE	주문 유형	ENUM_ORDER_TYPE
ORDER_STATE	주문 상태	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	주문 만료 시간	datetime
ORDER_TIME_DONE	주문 실행 또는 취소 시간	datetime
ORDER_TIME_SETUP_MSC	1970년 1월 1일 이후 실행 주문 시간(밀리초)	long
ORDER_TIME_DONE_MSC	1970년 1월 1일 이후 주문 실행/취소 시간(밀리초)	long
ORDER_TYPE_FILLING	주문 작성 유형	ENUM_ORDER_TYPE_FILLING
ORDER_TYPE_TIME	주문 수명	ENUM_ORDER_TYPE_TIME
ORDER_MAGIC	주문을 작성한 엑스퍼트 어드바이저의 ID(각 엑스퍼트 어드바이저가 고유 번호를 지정하도록 설계됨)	long
ORDER_REASON	주문 근거 또는 출처	ENUM_ORDER_REASON
ORDER_POSITION_ID	실행 즉시 순서로 설정된 포지션 식별자. 실행된 각 주문은 이미 존재하는	long

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	포지션을 열거나 수정하는 거래를 초대합니다. 정확히 이 위치의 식별자는 현재 실행된 순서로 설정됩니다.	
ORDER_POSITION_BY_ID	ORDER_TYPE_CLOSE_BY 주문에 사용되는 반대 위치 식별자	long

다음 함수의 경우: OrderGetDouble() 및 HistoryOrderGetDouble()

ENUM_ORDER_PROPERTY_DOUBLE

식별자	설명	유형
ORDER_VOLUME_INITIAL	주문 초기 볼륨	double
ORDER_VOLUME_CURRENT	주문 현재 볼륨	double
ORDER_PRICE_OPEN	주문에 지정된 가격	double
ORDER_SL	Stop Loss 값	double
ORDER_TP	Take Profit 값	double
ORDER_PRICE_CURRENT	주문 기호의 현재 가격	double
ORDER_PRICE_STOPLIMIT	StopLimit 주문의 제한 주문 가격	double

다음 함수의 경우: OrderGetString() 및 HistoryOrderGetString()

ENUM_ORDER_PROPERTY_STRING

식별자	설명	유형
ORDER_SYMBOL	주문의 기호	string
ORDER_COMMENT	주문 코멘트	string
ORDER_EXTERNAL_ID	외부 거래 시스템(거래소)의 주문 식별자	string

© 2000-2025, MetaQuotes Ltd.

OrderSend() 기능을 사용하여 거래 요청을 보낼 때 일부 작업에서 주문 유형을 표시해야 합니다. 주문 유형은 특수 구조 *유형* 필드(특수 구조 MqlTradeRequest) ENUM_ORDER_TYPE 열거 값을 숳락할 수 있습니다.

ENUM_ORDER_TYPE

식별자	설명
ORDER_TYPE_BUY	마켓 매수 주문
ORDER_TYPE_SELL	마켓 매도 주문
ORDER_TYPE_BUY_LIMIT	매수 제한 보류 주문
ORDER_TYPE_SELL_LIMIT	매도 주문 보류 주문
ORDER_TYPE_BUY_STOP	매수 중지 보류 주문
ORDER_TYPE_SELL_STOP	매도 중지 보류 주문
ORDER_TYPE_BUY_STOP_LIMIT	주문 가격에 도달하면 보류 중인 매수 한도 주문이 StopLimit 가격에 배치됩니다
ORDER_TYPE_SELL_STOP_LIMIT	주문 가격에 도달하면 보류 중인 매도 제한 주문이 Stoplimit 가격에 배치됩니다
ORDER_TYPE_CLOSE_BY	반대 포지션으로 포지션을 닫도록 주문

각 주문에는 상태를 설명하는 상태가 있습니다. 정보를 얻으려면 OrderGetInteger() 또는 HistoryOrderGetInteger()를 (ORDER_STATE 수정자와 함께) 사용하십시오. 허용되는 값은 ENUM_ORDER_STATE 열거에 저장됩니다.

ENUM_ORDER_STATE

식별자	설명
ORDER_STATE_STARTED	주문을 확인했지만 브로커가 아직 수락하지 않았습니다
ORDER_STATE_PLACED	주문 수락
ORDER_STATE_CANCELED	클라이언트가 주문 취소
ORDER_STATE_PARTIAL	주문 부분 실행
ORDER_STATE_FILLED	주문 실행
ORDER_STATE_REJECTED	주문 거절
ORDER_STATE_EXPIRED	주문 만료
ORDER_STATE_REQUEST_ADD	주문 등록 중(트레이딩 시스템에 배치)
ORDER_STATE_REQUEST_MODIFY	주문 수정 중(매개변수 변경)
ORDER_STATE_REQUEST_CANCEL	주문 삭제 중(트레이딩 시스템에서 삭제)

© 2000-2025, MetaQuotes Ltd.

OrderSend() 함수를 사용하여 거래 요청을 보낼 때 *type_filling* 필드(특수 구조MqlTradeRequest)에 주문에 대한 채움 정책을 설정할 수 있습니다. ENUM_ORDER_TYPE_FILLING 열거 값이 허용됩니다. 이 속성 값을 얻으려면 OrderGetInteger() 또는 HistoryOrderGetInteger() 함수를 ORDER_TYPE_FILLING 수정자와 함께 사용하십시오.

ENUM_ORDER_TYPE_FILLING

식별자	설명
ORDER_FILLING_FOK	이 채움 정책은 지정된 양으로만 주문이 작성될 수 있음을 의미합니다. 현재 시장에서 금융 상품의 필요 금액이 없는 경우 주문이 실행되지 않습니다. 현재 시장에서 제공되는 몇 가지 제안을 사용하여 필요한 수량을 채울 수 있습니다.
ORDER_FILLING_IOC	이 모드는 트레이더가 주문서에 표시된 범위 내에서 시장에서 최대 가능 물량으로 거래를 실행하는데 동의함을 의미합니다. 전체 주문 물량을 채울 수 없는 경우, 사용 가능한 물량이 채워지고 나머지 물량은 취소됩니다.
Passive (Book or Cancel)	BoC 주문은 주문이 시장 심도에만 배치될 수 있고 즉시 실행될 수 없다고 가정합니다. 주문이 주문을 한 즉시 실행될 수 있는 경우 주문은 취소됩니다. 실제로 BOC 정책은 접수된 주문의 가격이 현재 시장보다 더 나쁠 것이라고 단언합니다. BoC 주문은 패시브 트레이딩을 구현하는 데 사용되므로 주문이 즉시 실행되지 않고 현재 유동성에 영향을 미치지 않습니다. 리밋 및 스탑리밋 주문만 지원됩니다 (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT, ORDER_TYPE_SELL_STOP_LIMIT).
ORDER_FILLING_RETURN	해당 정책은 시장 주문(ORDER_TYPE_BUY 및 ORDER_TYPE_SELL), 제한 및 중지 제한 주문 (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT 및 ORDER_TYPE_SELL_STOP_LIMIT)에 대해서만 사용되며, 마켓 또는 거래소 실행이 포함된 기호에만 사용됩니다. 시장 일부 충당이나 잔여 물량으로 한정 주문은 취소되지 않고 추가 처리됩니다. ORDER_TYPE_BUY_STOP_LIMIT 및 ORDER_TYPE_SELL_STOP_LIMIT 주문을 활성화하기 위해 ORDER_FILLING_RETURN 실행 유형을 가진 해당 제한 주문 ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT이 생성됩니다.

거래 요청을 보낼 때 *type_time* 필드(특수 구조 MqlTradeRequest)에서 OrderSend() 함수를 사용하여 설정할 수 있습니다. ENUM_ORDER_TYPE_TIME 열거 값이 허용됩니다. 이 속성 값을 얻으려면 ORDER_TYPE_TIME 수정자와 함께 OrderGetInteger() 또는 HistoryOrderGetInteger() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

ENUM_ORDER_TYPE_TIME

식별자	설명
ORDER_TIME_GTC	주문 취소 시까지 유효
ORDER_TIME_DAY	현재 거래일 주문까지 유효
ORDER_TIME_SPECIFIED	주문 만료 시까지 유효
ORDER_TIME_SPECIFIED_DAY	주문은 지정된 날짜의 23:59:59까지 유효합니다. 이 시간이 트레이딩 세션 외부에 있는 경우 주문이 가장 가까운 트레이딩 시간 내에 만료됩니다.

주문 발주 근거는 ORDER_REASON 속성에 포함되어 있습니다. StopOut 등의 결과로 모바일 애플리케이션에서 MQL5 프로그램으로 주문할 수 있습니다. 가능한 ORDER_REASON 값은 ENUM_ORDER_REASON 열거에 설명되어 있습니다.

ENUM_ORDER_REASON

식별자	설명
ORDER_REASON_CLIENT	데스크탑 터미널에서 주문이 작성되었습니다
ORDER_REASON_MOBILE	모바일 애플리케이션에서 주문을 발주했습니다
ORDER_REASON_WEB	주문이 웹 플랫폼에서 작성되었습니다
ORDER_REASON_EXPERT	주문은 MQL5 프로그램(예: 엑스퍼트 어드바이저 또는 스크립트)에서 작성되었습니다
ORDER_REASON_SL	Stop Loss 활성화의 결과로 주문이 작성되었습니다
ORDER_REASON_TP	Take Profit 활성화의 결과로 주문이 작성되었습니다
ORDER_REASON_SO	Stop Out 이벤트의 결과로 주문이 작성되었습니다

© 2000-2025, MetaQuotes Ltd.

포지션 속성

거래 작업을 수행하면 포지션이 열리거나, 그 볼륨 및/또는 방향이 변경되거나 사라집니다. 거래 작업은 주문에 따라 OrderSend() 기능이 거래 요청 형태로 발송되어 진행됩니다. 각 재무 보안 (기호)에 대해 하나의 열린 포지션만 사용할 수 있습니다. 포지션에는 PositionGet...() 함수로 읽을 수 있는 속성 집합이 있습니다.

다음 함수의 경우: PositionGetInteger()

ENUM_POSITION_PROPERTY_INTEGER

식별자	설명	유형
POSITION_TICKET	포지션 티켓. 새로 연 각 위치에 고유 번호가 할당되었습니다. 서버의 서비스 작업으로 인해 티켓이 변경된 경우를 제외하고 일반적으로 위치를 여는 데 사용되는 주문의 티켓과 일치합니다(예: 위치가 다시 열리는 스왑 충전). 위치를 여는 데 사용되는 순서를 찾으려면 POSITION_IDENTIFIER 속성을 적용합니다. POSITION_TICKET 값은 MqlTradeRequest::position에 해당합니다.	long
POSITION_TIME	포지션 공개 시간	datetime
POSITION_TIME_MSC	포지션 열림 시간(밀리초)은 1970년 1월 1일 이후입니다	long
POSITION_TIME_UPDATE	포지션 변경 시간	datetime
POSITION_TIME_UPDATE_MSC	위치 변경 시간(밀리초)은 1970년 1월 1일 이후입니다	long
POSITION_TYPE	포지션 유형	ENUM_POSITION_TYPE

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
POSITION_MAGIC	포지션 매직 넘버 (ORDER_MAGIC 참조)	long
POSITION_IDENTIFIER	포지션 식별자는 다시 열린 각 포지션에 할당된 고유 번호입니다. 그것은 수명 주기 동안 변하지 않으며, 포시션을 열기 위해 사용되는 주문의 티켓에 해당합니다. 포지션 식별자는 열기, 수정 또는 닫기에 사용되는 각 주문 (ORDER_POSITION_ID) 및 거래 (DEAL_POSITION_ID)로 지정됩니다. 이 속성을 사용하여 포지션과 관련된 주문 및 거래를 검색할 수 있습니다. 네팅(Netting) 모드에서 포지션을 반전할 때(단일 안/밖 트레이드를 사용), POSITION_IDENTIFIER는 변경되지 않습니다. 그러나 POSITION_TICKET은 반전을 초래한 주문의 티켓으로 대체됩니다. 위험 회피 모드에서 포지션 반전은 제공되지 않습니다.	long
POSITION_REASON	포지션 오프닝 이유	ENUM_POSITION_REASON

다음 함수의 경우: PositionGetDouble()

© 2000-2025, MetaQuotes Ltd.

ENUM_POSITION_PROPERTY_DOUBLE

식별자	설명	유형
POSITION_VOLUME	포지션 볼륨	double
POSITION_PRICE_OPEN	포지션 시가	double
POSITION_SL	열린 포지션의 Stop Loss 레벨	double
POSITION_TP	열린 포지션의 Take Profit 레벨	double
POSITION_PRICE_CURRENT	포지션 기호의 현재 가격	double
POSITION_SWAP	누적 스왑	double
POSITION_PROFIT	현재 수익	double

다음 함수의 경우: PositionGetString()

ENUM_POSITION_PROPERTY_STRING

식별자	설명	유형
POSITION_SYMBOL	포지션의 기호	string
POSITION_COMMENT	포지션 코멘트	string
POSITION_EXTERNAL_ID	외부 거래 시스템(거래소)에서 위치 식별자	string

열린 포지션(매수 또는 매도)의 방향은 ENUM_POSITION_TYPE 열거값으로 지정됩니다. 열린 포지션의 유형을 얻으려면 PositionGetInteger() 함수를 POSITION_TYPE 수정자와 함께 사용합니다.

ENUM_POSITION_TYPE

식별자	설명
POSITION_TYPE_BUY	매수
POSITION_TYPE_SELL	매도

포지션을 여는 이유는 POSITION_REASON 속성에 포함되어 있습니다. 데스크탑 터미널, 모바일 애플리케이션, 엑스퍼트 어드바이저 등에 의해 주문이 활성화된 경우 포지션을 열 수 있습니다. 가능한 POSITION_REASON 값은 ENUM_POSITION_REASON 열거에 설명되어 있습니다.

ENUM_POSITION_REASON

© 2000-2025, MetaQuotes Ltd.

식별자	설명
POSITION_REASON_CLIENT	데스크탑 터미널에서 주문이 활성화된 결과 포지션이 열렸습니다
POSITION_REASON_MOBILE	모바일 앱에서 주문이 활성화된 결과로 포지션이 열렸습니다
POSITION_REASON_WEB	웹 플랫폼에서 작성된 주문이 활성화되어 포지션이 열렸습니다
POSITION_REASON_EXPERT	이 포지션은 MQL5 프로그램(엑스퍼트 어드바이저 또는 스크립트)에서 작성된 주문의 활성화 결과로 개설되었습니다

© 2000-2025, MetaQuotes Ltd.

딜 속성

거래는 거래 작업 실행 사실을 거래 요청이 포함된 주문에 근거하여 반영한 것입니다. 각 거래는 정보를 얻을 수 있는 자산에 의해 설명됩니다. 속성 값을 읽기 위해 해당 열거에서 값을 반환하는 HistoryDealGet...() 유형 함수가 사용됩니다.

다음 함수의 경우: HistoryDealGetInteger()

ENUM_DEAL_PROPERTY_INTEGER

식별자	설명	유형
DEAL_TICKET	거래 티켓. 각 거래에 할당된 고유 번호	long
DEAL_ORDER	거래 주문 수	long
DEAL_TIME	거래 시간	datetime
DEAL_TIME_MSC	1970년 1월 1일 이후 거래 실행 시간(밀리초)	long
DEAL_TYPE	거래 유형	ENUM_DEAL_TYPE
DEAL_ENTRY	Deal entry - entry in, entry out, reverse	ENUM_DEAL_ENTRY
DEAL_MAGIC	딜 매직 넘버 (ORDER_MAGIC참조)	long
DEAL_REASON	거래 실행 근거 또는 출처	ENUM_DEAL_REASON
DEAL_POSITION_ID	다음 위치에 대한 식별자 : 해당 거래가 체결된 시작, 수정 또는 마감의 위치. 각 위치에는 해당 위치의 전체 수명 동안 기호에 대해 실행된 모든 거래에 할당되는 고유한 식별자가 있습니다.	long

다음 함수의 경우: HistoryDealGetDouble()

ENUM_DEAL_PROPERTY_DOUBLE

식별자	설명	유형
DEAL_VOLUME	딜 볼륨	double
DEAL_PRICE	딜 가격	double
DEAL_COMMISSION	딜 커미션	double
DEAL_SWAP	마감 시 누적 스왑	double
DEAL_PROFIT	딜 수익	double
DEAL_FEE	거래 수행 후 즉시 청구되는 거래 수수료	double
DEAL_SL	Stop Loss level • 진입 및 청산후 반대 방향 진입 거래는 포지션이 진입하거나 반대로 진입한 원래 주문	double

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
	의 스탑로스 값을 사용합니다. • 청산 거래는 포지션 청산 시와 같은 스탑로스를 사용합니다.	
DEAL_TP	Take Profit level • 진입 및 청산 후 반대 방향 진입 거래는 포지션이 열리거나 취소된 원래 주문의 값의 익절 값을 사용합니다 • 청산 거래는 포지션 청산 시점의 포지션 가치의 익절값을 사용합니다.	double

다음 함수의 경우: HistoryDealGetString()

ENUM_DEAL_PROPERTY_STRING

식별자	설명	유형
DEAL_SYMBOL	딜 기호	string
DEAL_COMMENT	딜 코멘트	string
DEAL_EXTERNAL_ID	외부 거래 시스템(거래소)의 거래 식별자	string

각 거래의 특성은 유형으로, 허용되는 값은 ENUM_DEAL_TYPE에 열거됩니다. 거래 유형에 대한 정보를 얻으려면 HistoryDealGetInteger() 함수를 DEAL_TYPE 수정자와 함께 사용하십시오.

ENUM_DEAL_TYPE

식별자	설명
DEAL_TYPE_BUY	매수
DEAL_TYPE_SELL	매도
DEAL_TYPE_BALANCE	잔고
DEAL_TYPE_CREDIT	신용
DEAL_TYPE_CHARGE	추가 과금
DEAL_TYPE_CORRECTION	보정
DEAL_TYPE_BONUS	보너스
DEAL_TYPE_COMMISSION	추가 커미션
DEAL_TYPE_COMMISSION_DAILY	일별 커미션

© 2000-2025, MetaQuotes Ltd.

식별자	설명
DEAL_TYPE_COMMISSION_MONTHLY	월별 커미션
DEAL_TYPE_COMMISSION_AGENT_DAILY	일별 에이전트 커미션
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	월별 에이전트 커미션
DEAL_TYPE_INTEREST	이자율
DEAL_TYPE_BUY_CANCELED	매수 거래 취소. 이전에 진행한 매수 거래가 취소되는 상황이 발생할 수 있습니다. 이 경우 이전에 실행된 거래 유형(DEAL_TYPE_BUY)이 DEAL_TYPE_BUY_CANCELED로 변경되었으며 손익/손실이 0이 됩니다. 별도 잔액 작업을 사용하여 존속 이익/손실이 청구/인출됩니다
DEAL_TYPE_SELL_CANCELED	매도 거래 취소. 이전에 실행된 판매 거래가 취소되는 상황이 발생할 수 있습니다. 이 경우 이전에 실행된 거래 유형(DEAL_TYPE_SELL)이 DEAL_TYPE_SELL_CANCELED로 변경되었으며 손익/손실이 0이 됩니다. 별도 잔액 작업을 사용하여 존속 이익/손실이 청구/인출됩니다
DEAL_DIVIDEND	배당 사업
DEAL_DIVIDEND_FRANKED	프랭크(비과세) 배당 사업
DEAL_TAX	세금 부과

거래는 ENUM_DEAL_TYPE에 설정된 유형뿐만 아니라 위치 변경 방식에도 차이가 있습니다. 이는 간단한 위치 개방 또는 이전에 열린 위치 누적(시장 진입), 해당 볼륨(시장 이탈)의 반대 딜에 의해 포지션 폐쇄 또는 반대 방향 딜이 이전에 열린 포지션의 볼륨을 커버하는 경우 포지션 후진일 수 있습니다.

이러한 모든 상황은 ENUM_DEAL_ENTRY 열거값으로 설명됩니다. 거래에 대한 이 정보를 받으려면 HistoryDealGetInteger() 함수를 DEAL_ENTRY 수정자와 함께 사용하십시오.

ENUM_DEAL_ENTRY

식별자	설명
DEAL_ENTRY_IN	엔트리 인
DEAL_ENTRY_OUT	엔트리 아웃
DEAL_ENTRY_INOUT	역방향
DEAL_ENTRY_OUT_BY	반대 포지션으로 포지션 닫기

딜 실행 근거는 DEAL_REASON 속성에 포함되어 있습니다. 거래는 모바일 애플리케이션 또는 MQL5 프로그램에서 배치된 주문의 트리거링 및 StopOut 이벤트, 변동 마진 계산 등의 결과로 실행될 수 있습니다. 가능한 DEAL_REASON 값은 ENUM_DEAL_REASON 열거에 설명되어 있습니다. 잔고, 신용, 수수료 및 기타 영업에서 발생하는 비거래 딜의 경우 DEAL_REASON_CLIENT가 그 근거로 표시됩니다.

ENUM_DEAL_REASON

© 2000-2025, MetaQuotes Ltd.

식별자	설명
DEAL_REASON_CLIENT	데스크탑 터미널에서 주문이 활성화된 결과 거래가 실행되었습니다
DEAL_REASON_MOBILE	모바일 애플리케이션에서 주문이 활성화된 결과 거래가 실행되었습니다
DEAL_REASON_WEB	웹 플랫폼에서 작성된 주문이 활성화되어 거래가 실행되었습니다
DEAL_REASON_EXPERT	MQL5 프로그램(엑스퍼트 어드바이저 또는 스크립트)에서 주문이 활성화된 결과 거래가 실행되었습니다
DEAL_REASON_SL	Stop Loss 활성화의 결과로 거래가 실행되었습니다
DEAL_REASON_TP	Take Profit 활성화의 결과로 거래가 실행되었습니다
DEAL_REASON_SO	Stop Out 이벤트의 결과로 거래가 실행되었습니다
DEAL_REASON_ROLLOVER	롤오버로 인해 거래가 실행되었습니다
DEAL_REASON_VMARGIN	변동 마진을 청구한 후 거래가 실행되었습니다
DEAL_REASON_SPLIT	분할 발표 시 개방형 포지션이었던 금융상품의 분할(가격 인하) 이후 거래가 실행되었습니다
DEAL_REASON_CORPORATE_ACTION	해당 거래는 증권 병합 또는 증권의 이름 변경, 고객을 다른 계좌로 이전 하는 등 기업 경영 활동의 결과로 실행되었습니다

© 2000-2025, MetaQuotes Ltd.

거래 작업 유형

거래는 OrderSend() 기능을 사용하여 열린 포지션으로 주문을 보내고 보류중인 주문을 작성, 수정 또는 삭제하는 방식으로 이루어집니다. 각 거래 주문은 요청된 작업의 유형을 나타냅니다. 거래 작업은 ENUM_TRADE_REQUEST_ACTIONS 열거에 설명되어 있습니다.

ENUM_TRADE_REQUEST_ACTIONS

식별자	설명
TRADE_ACTION_DEAL	지정된 매개변수를 사용하여 즉시 실행할 거래 주문 작성(시장 주문)
TRADE_ACTION_PENDING	지정된 조건에서 실행에 대한 거래 주문 작성(보류 중인 주문)
TRADE_ACTION_SLTP	열린 포지션의 Stop Loss 및 Take Profit 값
TRADE_ACTION_MODIFY	이전에 발주된 주문의 매개변수 수정
TRADE_ACTION_REMOVE	이전에 발주된 보류 중인 주문을 삭제
TRADE_ACTION_CLOSE_BY	반대 포지션으로 포지션 닫기

매입 포지션을 열기 위한 TRADE_ACTION_DEAL 거래 작업의 예:

```
#define EXPERT_MAGIC 123456   // MagicNumber of the expert
//+------------------------------------------------------------------+
//| 매입 위치 열기                                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
//--- 요청의 매개변수
   request.action   =TRADE_ACTION_DEAL;                     // 거래 작업 유형
   request.symbol   =Symbol();                              // symbol
   request.volume   =0.1;                                   // 0.1 로트의 볼륨
   request.type     =ORDER_TYPE_BUY;                        // 주문 유형
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // 개장 가격
   request.deviation=5;                                     // 가격에 대한 허용 편차
   request.magic    =EXPERT_MAGIC;                          // 주문의 MagicNumber
//--- 요청 전송
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // if unable to send the r
//--- 작업에 대한 정보
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

매도 포지션을 열기 위한 TRADE_ACTION_DEAL 거래 작업의 예:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // 엑스퍼트의 Magi cNumber
//+------------------------------------------------------------------+
//| 매도 위치 열기                                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
//--- 요청의 매개변수
   request.action   =TRADE_ACTION_DEAL;                     // 거래 작업 유형
   request.symbol   =Symbol();                              // symbol
   request.volume   =0.2;                                   // 0.2 로트의 볼륨
   request.type     =ORDER_TYPE_SELL;                       // 주문 유형
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // 개장 가격
   request.deviation=5;                                     // 가격에 대한 허용 편차
   request.magic    =EXPERT_MAGIC;                          // MagicNumber of the orde
//--- 요청 전송
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // if unable to send the r
//--- 작업에 대한 정보
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

마감 포지션에 대한 TRADE_ACTION_DEAL 거래 작업의 예:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // 엑스퍼트의 MagicNumber
//+------------------------------------------------------------------+
//| 모든 포지션 닫기                                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청 및 거래 요청 결과 선언하고 초기화
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 열린 포지션의 숫자
//--- 모든 열린 포지션에서 반복
   for(int i=total-1; i>=0; i--)
     {
      //--- 주문의 매개변수
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- 포지션에 대한 출력 정보
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- MagicNumber가 일치하면
      if(magic==EXPERT_MAGIC)
        {
         //--- 요청 및 결과 값 0 설정
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 작업 매개변수 설정
         request.action   =TRADE_ACTION_DEAL;        // 거래 작업의 유형
         request.position =position_ticket;          // t포지션의 티켓
         request.symbol   =position_symbol;          // 기호
         request.volume   =volume;                   // 포지션의 볼륨
         request.deviation=5;                        // 가격에서의 허용 편차
         request.magic    =EXPERT_MAGIC;             // 포지션의 MagicNumber
         //--- 포지션에 따라 가격과 주문의 종류를 설정
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- 마감에 대한 정보 출력
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToString
         //--- 요청 전송
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // 요청을 보낼 수 없는 경
         //--- 작업에 대한 정보
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
```

보류 중인 주문을 발주하기 위한 TRADE_ACTION_PENDING 거래 작업의 예:

© 2000-2025, MetaQuotes Ltd.

```
#property description "Example of placing pending orders"
#property script_show_inputs
#define EXPERT_MAGIC 123456                                  // MagicNumber of the expert
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // order type
//+------------------------------------------------------------------+
//| Placing pending orders                                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- declare and initialize the trade request and result of trade request
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
//--- parameters to place a pending order
   request.action   =TRADE_ACTION_PENDING;                                  // type of trad
   request.symbol   =Symbol();                                              // symbol
   request.volume   =0.1;                                                   // volume of 0.
   request.deviation=2;                                                     // allowed devi
   request.magic    =EXPERT_MAGIC;                                          // MagicNumber
   int offset = 50;                                                         // offset from
   double price;                                                            // order trigge
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);                     // value of poi
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);                     // number of de
   //--- checking the type of operation
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                               // order type
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;             // price for op
      request.price    =NormalizeDouble(price,digits);                      // normalized c
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                               // order type
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;              // price for c
      request.price    =NormalizeDouble(price,digits);                       // normalized
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                                    // order type
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;     // price for c
      request.price=NormalizeDouble(price,digits);                          // normalized
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                               // order type
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;             // price for c
      request.price    =NormalizeDouble(price,digits);                      // normalized
     }
   else Alert("This example is only for placing pending orders");   // if not pending
//--- send the request
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());                     // if unable t
//--- information about the operation
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

열린 포지션의 손실 중지 및 이익 취하기 값을 수정하기 위한 TRADE_ACTION_SLTP 거래 작업의 예:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // 엑스퍼트의 MagicNumber
//+------------------------------------------------------------------+
//| 손실 정지 수정 및 포지션 이익 취득                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 열린 포지션의 숫자
//--- 모든 열린 포지션에서 반복
   for(int i=0; i<total; i++)
     {
      //--- 주문의 매개변수
      ulong  position_ticket=PositionGetTicket(i);// 포지션의 티켓
      string position_symbol=PositionGetString(POSITION_SYMBOL); // 기호
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // 소수점 0
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // 포지션의 MagicNumber
      double volume=PositionGetDouble(POSITION_VOLUME);    // 포지션의 볼륨
      double sl=PositionGetDouble(POSITION_SL);  // 포지션의 손실 중지
      double tp=PositionGetDouble(POSITION_TP);  // 포지션의 이익 가져가기
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- 포지션에 대한 정보 출력
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- MagicNumber가 일치하는 경우 손실 중지 및 이익 취득이 정의되지 않습니다
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- 현재 가격 레벨 계산
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_L
         double price_level;
         //--- 현재 종가에서 허용되는 최소 오프셋 거리(포인트)가 설정되지 않은 경우
         if(stop_level<=0)
            stop_level=150; // 현재 종가 대비 오프셋 거리를 150 포인트로 설정
         else
            stop_level+=50; // set the offset distance to (SYMBOL_TRADE_STOPS_LEVEL +

         //--- Stop Loss 및 Take Profit 값의 계산 및 반올림
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(ask+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(bid-price_level,digits);
           }
         //--- 요청 및 결과 값 0 설정
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 작업 매개변수 설정
         request.action  =TRADE_ACTION_SLTP; // 거래 작업 유형
         request.position=position_ticket;   // 포지션의 티켓
         request.symbol=position_symbol;     // 기호
         request.sl      =sl;                // 포지션의 Stop Loss
         request.tp      =tp;                // 포지션의 Take Profit
         request.magic=EXPERT_MAGIC;         // 포지션의 매직넘버
         //--- 수정에 대한 정보 출력
         PrintFormat("Modify #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- 요청 전송
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // 요청을 보낼 수 없는 경
         //--- 작업 정보
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
   }
//+------------------------------------------------------------------+
```

보류 중인 주문의 가격 수준을 수정하기 위한 TRADE_ACTION_MODIFY 거래 작업의 예:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber of the expert
//+------------------------------------------------------------------+
//| 보류 중인 주문 수정                                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청의 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
   int total=OrdersTotal(); // 보류 중인 주문 총 수
//--- 보류 중인 주문을 전부 반복
   for(int i=0; i<total; i++)
     {
      //--- 주문의 매개변수
      ulong  order_ticket=OrderGetTicket(i);                                   // 주문 티켓
      string order_symbol=Symbol();                                            // 기호
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS);        // 소수점 이하
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                               // 주문의 매직
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);                      // 주문의 현재
      double sl=OrderGetDouble(ORDER_SL);                                      // 주문의 현재
      double tp=OrderGetDouble(ORDER_TP);                                      // 주문의 현재
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE);       // 주문의 유형
      int offset = 50;                                                         // 주문을 발주
      double price;                                                            // 가격 유발 ?
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);                // 포인트의 값
      //--- 주문 정보 출력
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- 매직넘버가 일치하는 경우 Stop Loss 및 Take Profit는 지정되지 않습니다
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                                   // 거래 작업 유형
         request.order = OrderGetTicket(i);                                    // 주문 티켓
         request.symbol   =Symbol();                                           // 기호
         request.deviation=5;                                                  // 가격 허용 편차
        //--- 주문의 유형에 따라 Take Profit 및 Stop Loss, 가격 수준 설정
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 정규 시
           }
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 정규 시
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

© 2000-2025, MetaQuotes Ltd.

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 정규 시
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 정규 시
           }
         //--- 요청 전송
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // 요청 전송을 할 수 없는
         //--- 작업 정보
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- 요청 및 결과 값 0 설정
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

보류 중인 주문을 삭제하기 위한 TRADE_ACTION_REMOVE 거래 작업의 예:

```
#define EXPERT_MAGIC 123456  // 엑스퍼트의 MagicNumber
//+------------------------------------------------------------------+
//| 보류 중인 주문 삭제                                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
   int total=OrdersTotal(); // 보류 중인 총 주문 수
//--- 보류 중인 주문을 전부 반복
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // 주문 티켓
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // 주문의 MagicNumber
      //--- MagicNumber가 일치하면
      if(magic==EXPERT_MAGIC)
        {
         //--- 요청 및 결과 값 0 설정
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 작업 매개변수 설정
         request.action=TRADE_ACTION_REMOVE;                   // 거래 작업 유형
         request.order = order_ticket;                         // 주문 티켓
         //--- 요청 전송
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // 요청을 보낼 수 없는 경
         //--- 작업 정보
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

서로 반대되는 포지션에 의한 포지션 폐쇄의 TRADE_ACTION_CLOSE_BY 거래 작업 예:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // MagicNumber of the expert
//+------------------------------------------------------------------+
//| 반대 포지션으로 모든 포지션 닫기                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 포지션 열기 수
//--- 모든 포지션 열기 반복
   for(int i=total-1; i>=0; i--)
     {
      //--- 주문의 매개변수
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      double sl=PositionGetDouble(POSITION_SL);
      double tp=PositionGetDouble(POSITION_TP);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- 포지션 정보 출력
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- 매직넘버가 일치하는 경우
      if(magic==EXPERT_MAGIC)
        {
         for(int j=0; j<i; j++)
           {
            string symbol=PositionGetSymbol(j); // 반대 포지션의 기호
            //--- 반대 포지션과 초기 포지션의 기호가 일치하는 경우
            if(symbol==position_symbol && PositionGetInteger(POSITION_MAGIC)==EXPERT_M
              {
               //--- 반대 포지션 유형 설정
               ENUM_POSITION_TYPE type_by=(ENUM_POSITION_TYPE)PositionGetInteger(POSIT
               //--- 처음 포지션과 반대 포지션의 유형이 일치할 경우 떠나기
               if(type==type_by)
                  continue;
               //--- 요청 및 결과 값 0
               ZeroMemory(request);
               ZeroMemory(result);
               //--- 작업 매개변수 설정
               request.action=TRADE_ACTION_CLOSE_BY;                          // 거래 작
               request.position=position_ticket;                              // 포지션
               request.position_by=PositionGetInteger(POSITION_TICKET);       // 반대 포
               //request.symbol     =position_symbol;
               request.magic=EXPERT_MAGIC;                                    // MagicN
               //--- 반대 포지션으로 폐쇄 정보를 출력
               PrintFormat("Close #%I64d %s %s by #%I64d",position_ticket,position_sym
               //--- 요청 전송
               if(!OrderSend(request,result))
                  PrintFormat("OrderSend error %d",GetLastError()); // 요청을 전송할 수
```

© 2000-2025, MetaQuotes Ltd.

```
                //--- 작업 정보
                PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result
               }
            }
         }
      }
   }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

거래 트랜잭션 유형

거래 계좌에 대해 몇 가지 명확한 작업을 수행하면 상태가 변경됩니다. 이러한 작업에는 다음이 포함됩니다:

- OrderSend 및 OrderSendAsync 기능과 추가 실행을 사용하여 클라이언트 터미널의 MQL5 응용 프로그램에서 거래 요청 전송;
- 터미널 그래픽 인터페이스 및 추가 실행을 통해 거래 요청 전송;
- 서버에서 주문 보류 및 주문 활성화 중지;
- 거래 서버측에서 작업 수행.

이러한 조치의 결과로 다음과 같은 매매 거래가 수행됩니다:

- 거래 요청 처리;
- 공개 주문 변경;
- 주문 내역 변경;
- 거래 내역 변경;
- 포지션 변경.

예를 들어, 시장 매수 주문을 보낼 때 처리되고, 계정에 대한 적절한 매수 주문이 생성되며, 주문이 실행되어 열린 주문 목록에서 삭제되고, 주문 기록에 추가되고, 적절한 거래가 내역에 추가되고 새로운 포지션이 생성됩니다. 이 모든 행위는 매매 거래입니다.

프로그래머가 거래 계정과 관련하여 수행된 작업을 추적할 수 있도록 OnTradeTransaction 기능이 제공되었습니다. 이 처리기를 사용하면 MQL5 애플리케이션의 계정에 적용되는 거래 트랜잭션을 가져올 수 있습니다. 거래 트랜잭션 설명은 MqlTradeTransaction 구조를 사용하여 OnTradeTransaction 첫 번째 매개변수에 제출됩니다.

거래 트랜잭션 유형은 MqlTradeTransaction 구조의 유형 매개변수에 제출됩니다. 가능한 거래 트랜잭션 유형은 다음 열거값으로 설명됩니다:

ENUM_TRADE_TRANSACTION_TYPE

식별자	설명
TRADE_TRANSACTION_ORDER_ADD	새 공개 주문을 추가.
TRADE_TRANSACTION_ORDER_UPDATE	공개 주문 업데이트. 업데이트에는 클라이언트 터미널 또는 거래 서버 측의 명백한 변경 사항뿐만 아니라 설정 시 주문 상태의 변경도 포함됩니다 (예: ORDER_STATE_STARTED에서 ORDER_STATE_PLACED로 또는 ORDER_STATE_PLACED에서 ORDER_STATE_PARTIAL로 등의 전환).
TRADE_TRANSACTION_ORDER_DELETE	공개 주문 목록에서 주문을 제거. 주문은 적절한 요청 또는 실행(작성)을 설정하고 내역으로 이동하면 공개 주문에서 삭제할 수 있습니다
TRADE_TRANSACTION_DEAL_ADD	내역에 거래 추가. 이 작업은 주문 실행 또는 계전 잔고로 작업 수행의 결과로 수행됩니다.
TRADE_TRANSACTION_DEAL_UPDATE	내역에 거래 업데이트. 이전에 실행된 거래가 서버에서 변경된 경우가 있을 수 있습니다. 예를 들어, 거래는 이

© 2000-2025, MetaQuotes Ltd.

식별자	설명
	전 브로커에 의해 이전되었던 외부 거래 시스템(거래소)에서 변경되었습니다.
TRADE_TRANSACTION_DEAL_DELETE	내역에서 거래 삭제. 이전에 실행된 거래가 서버에서 삭제되는 경우가 있을 수 있습니다. 예를 들어 거래에는 이전 브로커에 의해 이전되었던 외부 거래 시스템(거래소)에서 삭제되었습니다.
TRADE_TRANSACTION_HISTORY_ADD	실행 또는 취소 결과로 기록에 주문 추가.
TRADE_TRANSACTION_HISTORY_UPDATE	주문 내역에 있는 주문을 변경. 이 유형은 거래 서버 측의 기능을 강화하기 위해 제공됩니다.
TRADE_TRANSACTION_HISTORY_DELETE	거래 내역에서 주문 삭제. 이 유형은 거래 서버 측의 기능을 강화하기 위해 제공됩니다.
TRADE_TRANSACTION_POSITION	거래 실행과 관련없는 포지션 변경. 이 트랜잭션 유형은 거래 서버 쪽에서 포지션이 변경되었음을 나타냅니다. 포지션 볼륨, 공개 가격, Stop Loss 및 Take Profit 레벨을 변경할 수 있습니다. 변경 사항에 대한 데이터는 OnTradeTransaction 처리기를 통해 MqlTradeTransaction 구조로 제출됩니다. 거래 실행으로 인한 위치 변경(추가, 변경 또는 마감)으로 인해 TRADE_TRANSACTION_POSITION 트랜잭션이 발생하지 않습니다.
TRADE_TRANSACTION_REQUEST	거래요청이 서버에서 처리되어 처리 결과를 수신했습니다. MqlTradeTransaction 구조의 해당 트랜잭션에 대해 유형 필드(거래 트랜잭션 유형)만 분석해야 합니다. 추가 데이터를 위해 OnTradeTransaction의 두 번째 및 세 번째 매개변수(요청 및 결과)를 분석해야 합니다.

거래 트랜잭션 유형에 따라 MqlTradeTransaction 구조에 다양한 매개변수가 입력됩니다. 제출된 데이터에 대한 자세한 설명은 "거래 트랜잭션 구조"에 나와있습니다.

추가 참조

거래 트랜잭션 구조, OnTradeTransaction

© 2000-2025, MetaQuotes Ltd.

Depth Of Market의 거래 주문(Trade Orders)

지분 증권의 경우 현재 매수 및 매도 주문을 볼 수 있는 Depth of Market 창을 사용할 수 있습니다. 각 주문에 대해 원하는 거래 방향, 요구 금액 및 요청 가격이 명시되어 있습니다.

MQL5 수단으로 DOM의 현재 상태에 대한 정보를 얻으려면 MarketBookGet() 함수를 사용하여 DOM "스크린샷"을 MqlBookInfo 구조 배열에 배치합니다. *유형* 필드에 있는 배열의 각 요소에는 순서 방향(ENUM_BOOK_TYPE 열거값)에 대한 정보가 들어있습니다.

ENUM_BOOK_TYPE

식별자	설명
BOOK_TYPE_SELL	매도 주문 (제안)
BOOK_TYPE_BUY	매수 주문 (입찰)
BOOK_TYPE_SELL_MARKET	마켓별 매도 주문
BOOK_TYPE_BUY_MARKET	마켓별 매수 주문

추가 참조

구조 및 클래스, DOM 구조, 거래 작업 유형, 마켓 정보

© 2000-2025, MetaQuotes Ltd.

시그널 속성

이 포지션은 MQL5 프로그램(엑스퍼트 어드바이저 또는 스크립트)에서 작성된 주문의 활성화 결과로 개설되었습니다.

트레이딩 시그널의 double 유형 속성 열거:

ENUM_SIGNAL_BASE_DOUBLE

ID	설명
SIGNAL_BASE_BALANCE	계좌 잔고
SIGNAL_BASE_EQUITY	계좌 지분
SIGNAL_BASE_GAIN	계정 획득
SIGNAL_BASE_MAX_DRAWDOWN	계정 최대 인출액
SIGNAL_BASE_PRICE	시그널 구독 가격
SIGNAL_BASE_ROI	투자 수익률(%)

트레이딩 시그널의 integer 유형 속성 열거:

ENUM_SIGNAL_BASE_INTEGER

ID	설명
SIGNAL_BASE_DATE_PUBLISHED	등록일 (가입 가능 날짜)
SIGNAL_BASE_DATE_STARTED	모니터링 시작일
SIGNAL_BASE_DATE_UPDATED	신호의 거래 통계의 마지막 업데이트 날짜
SIGNAL_BASE_ID	시그널 ID
SIGNAL_BASE_LEVERAGE	계좌 레버리지
SIGNAL_BASE_PIPS	수익(pips 단위)
SIGNAL_BASE_RATING	등급에서 포지션
SIGNAL_BASE_SUBSCRIBERS	구독자 수
SIGNAL_BASE_TRADES	거래 수
SIGNAL_BASE_TRADE_MODE	계좌 유형 (0-실제, 1-데모, 2-컨테스트)

트레이딩 시그널의 string 유형 속성 열거:

ENUM_SIGNAL_BASE_STRING

ID	설명
SIGNAL_BASE_AUTHOR_LOGIN	작성자 로그인
SIGNAL_BASE_BROKER	브로커명(회사)

© 2000-2025, MetaQuotes Ltd.

ID	설명
SIGNAL_BASE_BROKER_SERVER	브로커 서버
SIGNAL_BASE_NAME	시그널 이름
SIGNAL_BASE_CURRENCY	시그널 기준 통화

시그널 복사 설정의 double 유형 특성 열거:

ENUM_SIGNAL_INFO_DOUBLE

ID	설명
SIGNAL_INFO_EQUITY_LIMIT	지분 한도
SIGNAL_INFO_SLIPPAGE	슬리페이지(포지션 동기화 및 거래 복사 시 시장 주문을 작성할 때 사용)
SIGNAL_INFO_VOLUME_PERCENT	사용한 예치금의 최대 백분율 (%), r/o

시그널 복사 설정의 integer 유형 속성 열거:

ENUM_SIGNAL_INFO_INTEGER

ID	설명
SIGNAL_INFO_CONFIRMATIONS_DISABLED	플래그가 확인 없이 동기화를 활성화합니다
SIGNAL_INFO_COPY_SLTP	Stop Loss 및 Take Profit 플래그 복사
SIGNAL_INFO_DEPOSIT_PERCENT	예치율 (%)
SIGNAL_INFO_ID	시그널 ID, r/o
SIGNAL_INFO_SUBSCRIPTION_ENABLED	"구독별 거래 복사" 권한 플래그
SIGNAL_INFO_TERMS_AGREE	"시그널 서비스 이용 약관에 동의" 플래그, r/o

시그널 복사 설정의 string 유형 속성 열거:

ENUM_SIGNAL_INFO_STRING

ID	설명
SIGNAL_INFO_NAME	시그널 이름, r/o

추가 참조

거래 신호

© 2000-2025, MetaQuotes Ltd.

명명된 상수

MQL5에서 사용되는 모든 상수는 다음과 같은 그룹으로 나눌 수 있습니다:

- 사전 정의된 매크로 대체 – 컴파일 중에 값이 대체됩니다;
- 수학 상수 – 일부 수학 표현식의 값;
- 숫자 유형 상수 – 간단한 유형 제한사항 중 일부;
- 초기화 취소 사유 코드 – 초기화 취소 사유 설명;
- 개체 포인터 확인 - CheckPointer() 함수에 의해 반환된 포인터 유형의 열거;
- 기타 상수 – 기타 모든 상수.

© 2000-2025, MetaQuotes Ltd.

사전 정의된 매크로 대체

디버깅 프로세스를 단순화하고 mql5 프로그램의 작동에 대한 정보를 얻기 위해 컴파일 시 설정되는 특수 매크로 상수가 있습니다. 이 상수를 사용하는 가장 쉬운 방법은 예제와 같이 Print() 함수로 값을 출력하는 것입니다.

정수	설명
__CPU_ARCHITECTURE__	컴파일된 아키텍처 이름(명령 집합) EX5 파일
__DATE__	시간이 없는 파일 컴파일 날짜(시간, 분 및 초는 0)
__DATETIME__	파일 컴파일 날짜 및 시간
__LINE__	매크로가 위치한 소스 코드의 라인 번호
__FILE__	현재 컴파일된 파일의 이름
__PATH__	현재 컴파일되고 있는 파일의 절대 경로입니다
__FUNCTION__	매크로가 위치한 본문의 함수 이름
__FUNCSIG__	매크로가 위치한 본문의 함수 서명입니다. 기능의 전체 설명을 로깅하는 것은 과부하된 기능 식별에 유용할 수 있습니다.
__MQLBUILD__,__MQL5BUILD__	컴파일러 빌드 번호
__COUNTER__	The compiler for each encountered __COUNTER__ declaration substitutes the counter value from 0 to N-1 where N is a number of uses in the code. The __COUNTER__ order is guaranteed when recompiling the source code with no changes. The __COUNTER__ value is calculated the following way: • the initial counter value is 0, • after each counter usage, its value is increased by 1, • first, the compiler expands all macros and templates into source code on-site, • a separate code is created for each template function specialization, • a separate code is created for each template class/structure specialization, • next, the compiler passes through the obtained source code in the defined order and replaces each detected __COUNTER__ usage with the current counter value. The example below shows how the compiler handles the source code and replaces all instances of __COUNTER__ it meets with sequentially increasing values.
__RANDOM__	The compiler inserts a random ulong value for each __RANDOM__ declaration.

예:

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 기능                                             |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 엑스퍼트 어드바이저 초기화 시 정보 출력 예
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- 타이머 이벤트 간 간격 설정
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 취소 함수                                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 엑스퍼트 어드바이저 초기화 취소 시 정보 출력의 예
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 기능                                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 틱 수신 시 정보 출력
   Print(" __MQLBUILD__ = ",__MQLBUILD__,"  __FILE__ = ",__FILE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------+
//| test1                                                            |
//+------------------------------------------------------------------+
void test1(string par)
  {
//--- 함수 내 정보 출력
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par = ",par);
  }
//+------------------------------------------------------------------+
//| test2                                                            |
//+------------------------------------------------------------------+
void test2()
  {
//--- 함수 내 정보 출력
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| OnTimer 이벤트 처리기                                            |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

The example for learning how to work with the __COUNTER__ macro

```
//--- create a macro for a quick display of the expression and its value in the journa
#define print(expr) Print(#expr,"=",expr)

//--- define the MACRO_COUNTER custom macro via the predefined __COUNTER__ macro
#define MACRO_COUNTER __COUNTER__

//--- set the input value of the variable using the __COUNTER__ macro
input int InpVariable = __COUNTER__;

//--- set the value of the global variable using the __COUNTER__ macro before defining
int ExtVariable = __COUNTER__;

//+------------------------------------------------------------------+
//| the function returns the __COUNTER__ value                       |
//+------------------------------------------------------------------+
int GlobalFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| the template function returns the __COUNTER__ value              |
//+------------------------------------------------------------------+
template<typename T>
int GlobalTemplateFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| the structure with the method returning __COUNTER__              |
//+------------------------------------------------------------------+
struct A
  {
   int               dummy;  // not used

   int               Method(void)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       return(__COUNTER__);
      }
  };
//+------------------------------------------------------------------+
//| the template structure with the method returning __COUNTER__     |
//+------------------------------------------------------------------+
template<typename T>
struct B
  {
   int               dummy;  // not used

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| the structure with the template method returning __COUNTER__     |
//+------------------------------------------------------------------+
struct C
  {
   int               dummy;  // not used

   template<typename T>
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| the function #2, which returns the __COUNTER__ value             |
//+------------------------------------------------------------------+
int GlobalFunc2(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart(void)
  {
// __COUNTER__ in the macro and the variables
   print(MACRO_COUNTER);
   print(InpVariable);
   print(ExtVariable);

//--- __COUNTER__ in the functions
   print(GlobalFunc());
```

© 2000-2025, MetaQuotes Ltd.

```
   print(GlobalFunc());                 // the value is not changed
   print(GlobalTemplateFunc<int>());
   print(GlobalTemplateFunc<int>());    // the value is not changed
   print(GlobalTemplateFunc<double>());// the value has changed
   print(GlobalFunc2());
   print(GlobalFunc2());                // the value is not changed

// __COUNTER__ in the structure
   A a1, a2;
   print(a1.Method());
   print(a2.Method());                  // the value is not changed

// __COUNTER__ in the template structure
   B<int> b1, b2;
   B<double> b3;
   print(b1.Method());
   print(b2.Method());                  // the value is not changed
   print(b3.Method());                  // the value has changed

// __COUNTER__ in the structure with the template function
   C c1, c2;
   print(c1.Method<int>());
   print(c1.Method<double>());          // the value has changed
   print(c2.Method<int>());             // the same value as during the first c1.Metho

//--- let's have another look at __COUNTER__ in the macro and the global variable
   print(MACRO_COUNTER);  // the value has changed
   print(ExtGlobal2);
  }
//--- set the value of the global variable using the __COUNTER__ macro after defining
int ExtGlobal2 = __COUNTER__;
//+------------------------------------------------------------------+

/* Result
   __COUNTER__=3
   InpVariable=0
   ExtVariable=1
   GlobalFunc()=5
   GlobalFunc()=5
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<int>()=8
   GlobalTemplateFunc<double>()=9
   GlobalFunc2()=7
   GlobalFunc2()=7
   a1.Method()=6
   a2.Method()=6
   b1.Method()=10
   b2.Method()=10
   b3.Method()=11
```

© 2000-2025, MetaQuotes Ltd.

```
   c1.Method<int>()=12
   c1.Method<double>()=13
   c2.Method<int>()=12
   __COUNTER__=4
   ExtGlobal2=2

*/
```

© 2000-2025, MetaQuotes Ltd.

수학 상수

값을 포함하는 특수 상수는 일부 수학 식에 예약됩니다. 이러한 상수는 수학 함수를 사용하여 계산하는 대신 프로그램의 모든 위치에서 사용할 수 있습니다.

정수	설명	값
M_E	e	2.71828182845904523536
M_LOG2E	log2(e)	1.44269504088896340736
M_LOG10E	log10(e)	0.43429448190325182765 1
M_LN2	ln(2)	0.69314718055994530941 7
M_LN10	ln(10)	2.30258509299404568402
M_PI	pi	3.14159265358979323846
M_PI_2	pi/2	1.57079632679489661923
M_PI_4	pi/4	0.78539816339744830961 6
M_1_PI	1/pi	0.31830988618379067153 8
M_2_PI	2/pi	0.63661977236758134307 6
M_2_SQRTPI	2/sqrt(pi)	1.12837916709551257390
M_SQRT2	sqrt(2)	1.41421356237309504880
M_SQRT1_2	1/sqrt(2)	0.70710678118654752440 1

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 상수 값을 인쇄
   Print("M_E = ",DoubleToString(M_E,16));
   Print("M_LOG2E = ",DoubleToString(M_LOG2E,16));
   Print("M_LOG10E = ",DoubleToString(M_LOG10E,16));
   Print("M_LN2 = ",DoubleToString(M_LN2,16));
   Print("M_LN10 = ",DoubleToString(M_LN10,16));
   Print("M_PI = ",DoubleToString(M_PI,16));
   Print("M_PI_2 = ",DoubleToString(M_PI_2,16));
   Print("M_PI_4 = ",DoubleToString(M_PI_4,16));
   Print("M_1_PI = ",DoubleToString(M_1_PI,16));
   Print("M_2_PI = ",DoubleToString(M_2_PI,16));
   Print("M_2_SQRTPI = ",DoubleToString(M_2_SQRTPI,16));
   Print("M_SQRT2 = ",DoubleToString(M_SQRT2,16));
   Print("M_SQRT1_2 = ",DoubleToString(M_SQRT1_2,16));
  }
```

© 2000-2025, MetaQuotes Ltd.

숫자 유형 상수

각 간단한 숫자 유형은 특정 유형의 작업에 사용되며 올바르게 사용할 경우 mql5 프로그램의 작동을 최적화할 수 있습니다. 더 나은 코드 가독성과 계산 결과의 올바른 처리를 위해 특정 유형의 단순 데이터로 설정된 제한에 대한 정보를 수신할 수 있는 상수가 있습니다.

정수	설명	값
CHAR_MIN	char 유형으로 나타낼 수 있는 최소값	-128
CHAR_MAX	char 유형으로 나타낼 수 있는 최대값	127
UCHAR_MAX	uchar 유형으로 나타낼 수 있는 최대값	255
SHORT_MIN	short 유형으로 나타낼 수 있는 최소값	-32768
SHORT_MAX	short 유형으로 나타낼 수 있는 최대값	32767
USHORT_MAX	ushort 유형으로 나타낼 수 있는 최대값	65535
INT_MIN	int 유형으로 나타낼 수 있는 최소값	-2147483648
INT_MAX	int 유형으로 나타낼 수 있는 최대값	2147483647
UINT_MAX	uint 유형으로 나타낼 수 있는 최대값	4294967295
LONG_MIN	long 유형으로 나타낼 수 있는 최소값	-9223372036854775808
LONG_MAX	long 유형으로 나타낼 수 있는 최대값	9223372036854775807
ULONG_MAX	ulong 유형으로 나타낼 수 있는 최대값	18446744073709551615
DBL_MIN	double 유형으로 나타낼 수 있는 최소 양수값	2.2250738585072014e-308
DBL_MAX	double 유형으로 나타나낼 수 있는 최대값	1.7976931348623158e+308
DBL_EPSILON	조건을 만족하는 최소값: 1.0+DBL_EPSILON != 1.0 (double 유형의 경우)	2.2204460492503131e-016
DBL_DIG	double 유형의 유효 소수 자릿수	15
DBL_MANT_DIG	double 유형에 대한 mantissa의 비트 수	53
DBL_MAX_10_EXP	double 유형에 대한 지수도의 최대 소수 값	308
DBL_MAX_EXP	double 유형에 대한 지수도의 최대 binary 값	1024
DBL_MIN_10_EXP	double 유형의 지수도 최소 소수 값	(-307)
DBL_MIN_EXP	double 유형에 대한 지수도의 최소 binary 값	(-1021)
FLT_MIN	float 유형으로 나타낼 수 있는 최소 양수값	1.175494351e-38
FLT_MAX	float 유형으로 나타낼 수 있는 최대값	3.402823466e+38

© 2000-2025, MetaQuotes Ltd.

정수	설명	값
FLT_EPSILON	조건을 만족하는 최소값: 1.0+DBL_EPSILON != 1.0 (float 유형의 경우)	1.192092896e-07
FLT_DIG	float 유형의 유효 소수 자릿수	6
FLT_MANT_DIG	float 유형에 대한 mantissa 비트 수	24
FLT_MAX_10_EXP	float 유형의 지수도 최대 소수 값	38
FLT_MAX_EXP	float 유형의 지수도 최대 binary 값	128
FLT_MIN_10_EXP	float 유형의 지수도 최소 소수값	-37
FLT_MIN_EXP	float 유형의 지수도 최소 binary 값	(-125)

예:

```
void OnStart()
  {
//--- 상수 값을 출력
   printf("CHAR_MIN = %d",CHAR_MIN);
   printf("CHAR_MAX = %d",CHAR_MAX);
   printf("UCHAR_MAX = %d",UCHAR_MAX);
   printf("SHORT_MIN = %d",SHORT_MIN);
   printf("SHORT_MAX = %d",SHORT_MAX);
   printf("USHORT_MAX = %d",USHORT_MAX);
   printf("INT_MIN = %d",INT_MIN);
   printf("INT_MAX = %d",INT_MAX);
   printf("UINT_MAX = %u",UINT_MAX);
   printf("LONG_MIN = %I64d",LONG_MIN);
   printf("LONG_MAX = %I64d",LONG_MAX);
   printf("ULONG_MAX = %I64u",ULONG_MAX);
   printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
   printf("DBL_MIN = %.16e",DBL_MIN);
   printf("DBL_MAX = %.16e",DBL_MAX);
   printf("DBL_EPSILON = %.16e",DBL_EPSILON);
   printf("DBL_DIG = %d",DBL_DIG);
   printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
   printf("DBL_MAX_10_EXP =  %d",DBL_MAX_10_EXP);
   printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
   printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
   printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
   printf("FLT_MIN = %.8e",FLT_MIN);
   printf("FLT_MAX = %.8e",FLT_MAX);
   printf("FLT_EPSILON = %.8e",FLT_EPSILON);
/*
   CHAR_MIN = -128
```

© 2000-2025, MetaQuotes Ltd.

```
   CHAR_MAX = 127
   UCHAR_MAX = 255
   SHORT_MIN = -32768
   SHORT_MAX = 32767
   USHORT_MAX = 65535
   INT_MIN = -2147483648
   INT_MAX = 2147483647
   UINT_MAX = 4294967295
   LONG_MIN = -9223372036854775808
   LONG_MAX = 9223372036854775807
   ULONG_MAX = 18446744073709551615
   EMPTY_VALUE = 1.7976931348623157e+308
   DBL_MIN = 2.2250738585072014e-308
   DBL_MAX = 1.7976931348623157e+308
   DBL_EPSILON = 2.2204460492503131e-16
   DBL_DIG = 15
   DBL_MANT_DIG = 53
   DBL_MAX_10_EXP =  308
   DBL_MAX_EXP = 1024
   DBL_MIN_10_EXP = -307
   DBL_MIN_EXP = -1021
   FLT_MIN = 1.17549435e-38
   FLT_MAX = 3.40282347e+38
   FLT_EPSILON = 1.19209290e-07
*/
  }
```

© 2000-2025, MetaQuotes Ltd.

초기화 취소 이유 코드

초기화 취소 이유 코드는 UninitializeReason() 함수에 의해 반환됩니다. 가능 값은 다음과 같습니다:

정수	값	설명
REASON_PROGRAM	0	엑스퍼트 어드바이저가 ExpertRemove() 기능을 호ㅜㄹ하여 작업을 종료했습니다
REASON_REMOVE	1	프로그램이 차트에서 삭제되었습니다
REASON_RECOMPILE	2	프로그램이 다시 컴파일되었습니다
REASON_CHARTCHANGE	3	기호 또는 차트 기간이 변경되었습니다
REASON_CHARTCLOSE	4	차트가 닫혔습니다
REASON_PARAMETERS	5	입력 매개변수가 사용자에 의해 변경되었습니다
REASON_ACCOUNT	6	계정 설정 변경으로 인해 다른 계정이 활성화되었거나 거래 서버에 다시 연결되었습니다
REASON_TEMPLATE	7	새 템플릿이 적용되었습니다
REASON_INITFAILED	8	이 값은 OnInit() 처리기가 0이 아닌 값을 반환했음을 의미합니다
REASON_CLOSE	9	터미널이 닫혔습니다

초기화 취소 이유 코드는 미리 결정된 함수 OnDeinit(일시적 이유)의 매개변수로도 전달됩니다.

예:

```
//+------------------------------------------------------------------+
//| 텍스트 설명 가져오기                                             |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="계좌가 변경되었습니다";break;
      case REASON_CHARTCHANGE:
         text="기호 또는 시간 프레임이 변경되었습니다";break;
      case REASON_CHARTCLOSE:
         text="차트가 닫혔습니다";break;
      case REASON_PARAMETERS:
         text="입력 매개변수가 변경되었습니다";break;
      case REASON_RECOMPILE:
         text="Program "+__FILE__+" 다시 컴파일 되었습니다";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" 차트에서 삭제되었습니다";break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case REASON_TEMPLATE:
         text="차트에 새 템플릿이 적용되었습니다";break;
      default:text="다른 이유";
     }
//---
   return text;
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 취소 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 초기화 취소 이유 코드를 가져오는 첫 번째 방법
   Print(__FUNCTION__,"_Uninitalization reason code = ",reason);
   //--- 초기화 취소 이유 코드를 가져오는 두 번째 방법
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

© 2000-2025, MetaQuotes Ltd.

개체 포인터 확인

CheckPointer() 함수는 개체 포인터의 유형을 확인하는 데 사용됩니다. 함수는 ENUM_POINTER_TYPE 열거의 값을 반환합니다. 잘못된 포인터를 사용하면 프로그램 실행이 즉시 종료됩니다.

new() 연산자가 만든 개체는 POINTER_DYNAMIC 유형입니다. delete() 연산자 이러한 포인터에만 사용할 수 있으며 사용해야 합니다.

다른 모든 포인터는 POINTER_AUTOMATIC 유형이며, 이는 이 개체가 mql5 프로그램 환경에 의해 자동으로 생성되었으므을 의미합니다. 이러한 개체는 사용 후 자동으로 삭제됩니다.

ENUM_POINTER_TYPE

정수	설명
POINTER_INVALID	잘못된 포인터
POINTER_DYNAMIC	new() 연산자가 만든 개체 포인터
POINTER_AUTOMATIC	자동으로 생성된 개체의 포인터 (new() 사용하지 않음)

추가 참조

런타임 오류, 개체 삭제 연산자 삭제, CheckPointer

© 2000-2025, MetaQuotes Ltd.

다른 상수

CLR_NONE 상수는 색상 부재이 윤곽을 그리는 데 사용되며, 이는 지표의 그래픽 개체 또는 그래픽 시리즈가 그려지지 않음을 의미합니다. 이 상수는 웹 색상 상수 목록에 포함되지 않았지만 색상 인수가 필요한 모든 곳에 적용할 수 있습니다.

INVALID_HANDLE 상수를 사용하여 파일 핸들을 확인할 수 있습니다 (FileOpen() 및 FileFindFirst() 참조).

정수	설명	값
CHARTS_MAX	터미널에서 동시에 열려 있는 최대 차트 수	100
clrNONE	색상 없음	-1
EMPTY_VALUE	지표 버퍼의 빈 값	DBL_MAX
INVALID_HANDLE	잘못된 핸들	-1
IS_DEBUG_MODE	mq5 프로그램이 디버그 모드에서 작동하는 플래그	디버그 모드에서는 0이 아님, 그렇지 않으면 0
IS_PROFILE_MODE	mq5 프로그램이 프로파일링 모드에서 작동하는 플래그	프로파일링 모드에서 0이 아님, 그렇지 않으면 0
NULL	모든 유형에 대해 0	0
WHOLE_ARRAY	배열이 끝날 때까지 남아 있는 항목 수를 의미합니다. 즉, 전체 배열이 처리됩니다.	-1
WRONG_VALUE	암시적으로 상수를 캐스팅할 수 있습니다 (모든 열거 유형에)	-1

일반적으로 EMPTY_VALUE 상수는 차트에 표시되지 않은 지표의 값에 해당합니다. 예를 들어 주기가 20인 기본 제공 지표 표준 편차의 경우 내역의 처음 19개 막대에 대한 선이 차트에 표시되지 않습니다. iStdDev() 함수로 이 지표의 핸들을 만든 후 CopyBuffer()를 통해 이러한 막대의 지표 값 배열에 복사하면 이 값은 EMPTY_VALUE와 같습니다.

차트에 지표를 그려서는 안되는 경우 사용자 지정 지표에 대해 지표의 빈 값을 지정하도록 선택할 수 있습니다. PlotIndexSetDouble() 함수를 PLOT_EMPTY_VALUE 한정자와 함께 사용합니다.

NULL 상수는 모든 단순 유형의 변수 또는 개체 구조 또는 클래스 포인터에 할당할 수 있습니다. 문자열 변수에 대한 NULL 할당은 이 변수의 전체 초기화 해제를 의미합니다.

WRONG_VALUE 상수는 열거 값을 반환해야 하는 경우에 사용하기 위한 상수로, 이 값은 잘못된 값이어야 합니다. 예를 들어 반환 값이 이 열거값의 값임을 알려야 하는 경우입니다. 이름으로 지정된 개체의 선 스타일을 반환하는 CheckLineStyle() 함수를 예로 들어보겠습니다. ObjectGetInteger()으로 스타일 검사 결과가 true이면 ENUM_LINE_STYLE의 값이 반환되고 그렇지 않으면 WRONG_VALUE가 반환됩니다.

```
void OnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("오류 라인 스타일 가져오기.");
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 이름으로 지정된 개체의 선 스타일을 반환                          |
//+------------------------------------------------------------------+
ENUM_LINE_STYLE CheckLineStyle(string name)
  {
   long style;
//---
   if(ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
      return((ENUM_LINE_STYLE)style);
   else
      return(WRONG_VALUE);
  }
```

WHOLE_ARRAY 상수는 처리된 배열의 요소 수를 지정해야 하는 함수용입니다:

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

지정된 위치에서 끝까지 모든 배열 값을 처리하도록 지정하려면 WHOLE_ARRAY 값만 지정해야 합니다.

IS_PROFILE_MODE 상수를 사용하면 프로파일 모드에서 올바른 데이터 수집을 위한 프로그램 작업을 변경할 수 있습니다. 프로파일링을 통해 개별 프로그램 조각(일반적으로 함수로 구성됨)의 실행 시간을 측정하고 해당 호출 수를 계산할 수 있습니다. 다음과 같이 프로파일링 모드에서 실행 시간을 결정하기 위해 Sleep() 함수 호출을 비활성화할 수 있습니다:

```
//--- 슬립이 프로파일링 결과에 큰 영향을 미칠 수 있음
if(!IS_PROFILE_MODE) Sleep(100); // 프로파일링 모드에서 Sleep() 호출 사용 안함
```

IS_PROFILE_MODE 상수 값은 컴파일러 도중 컴파일러에 의해 설정되고, 일반 모드에서는 0으로 설정됩니다. 프로파일링 모드에서 프로그램을 시작하면 특수 컴파일이 수행되고 IS_PROFILE_MODE 값이 0이 아닌 값으로 바뀝니다.

IS_DEBUG_MODE 상수는 디버깅 모드에서 mql5 프로그램의 작동을 약간 변경해야 할 때 유용할 수 있습니다. 예를 들어 디버그 모드에서 터미널 로그에 추가 디버깅 정보를 표시하거나 차트에 추가 그래픽 개체를 만들어야 할 수 있습니다.

다음은 Label 개체를 작성하고 스크립트 실행 모드에 따라 설명 및 색상을 설정하는 예입니다. MetaEditor에서 디버그 모드로 스크립트를 실행하려면 F5를 누릅니다. 터미널의 브라우저 창에서 스크립트를 실행하는 경우, 개체의 색상과 텍스트가 달라집니다.

예:

```
//+------------------------------------------------------------------+
//|                                            Check_DEBUG_MODE.mq5 |
//|                        Copyright © 2009, MetaQuotes Software Corp. |
//|                                             https://www.metaquotes.net |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Object",label_name,"not found. Error code = ",GetLastError());
      //--- 레이블 생성
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- X 좌표 설정
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- Y 좌표 설정
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // 디버깅 모드
        {
         //--- 스크립트 실행 모드에 대한 메시지 표시
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- 텍스트 색상을 빨간색으로 설정
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("색상을 설정할 수 없습니다. Error",GetLastError());
        }
      else               // 작동 모드
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
         //--- 텍스트 색상을 무색으로 설정
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("색상을 설정할 수 없습니다. Error ",GetLastError());
        }
      ChartRedraw();
      DebugBreak();    // 디버그 모드인 경우 종료가 발생합니다
     }
  }
```

암호화 방법

ENUM_CRYPT_METHOD 열거형은 CryptEncode() 및 CryptDecode() 함수에 사용되는 데이터 변환 방법을 지정하는 데 사용됩니다.

ENUM_CRYPT_METHOD

© 2000-2025, MetaQuotes Ltd.

정수	설명
CRYPT_BASE64	BASE64
CRYPT_AES128	128 비트 키 (16 바이트)를 사용한 AES 암호화
CRYPT_AES256	256 비트 키(32 바이트)를 사용한 AES 암호화
CRYPT_DES	56 비트 키(7 바이트)를 사용한 DES 암호화
CRYPT_HASH_SHA1	SHA1 HASH 계산
CRYPT_HASH_SHA256	SHA256 HASH 계산
CRYPT_HASH_MD5	MD5 HASH 계산
CRYPT_ARCH_ZIP	ZIP 아카이브

추가 참조

DebugBreak, 실행된 MQL5 프로그램 속성, CryptEncode(), CryptDecode()

© 2000-2025, MetaQuotes Ltd.

데이터 구조

MQL5 언어는 12개의 사전 정의된 구조를 제공합니다:

- MqlDateTime은 날짜 및 시간으로 작업하기 위한 것입니다;
- MqlParam이 IndicatorCreate() 함수를 사용하여 지표의 핸들을 만들 때 매개변수를 전송할 수 있습니다;
- MqlRates는 과거 데이터를 조작하기 위한 것이며 가격, 볼륨 및 스프레드에 대한 정보를 포함합니다;
- MqlBookInfo는 Depth of Market에 대한 정보를 얻기 위한 것입니다;
- MqlTradeRequest는 거래 작업에 대한 거래 요청을 생성하는 데 사용됩니다;
- MqlTradeCheckResult는 확인을 준비된 거래 요청을 전송하기 전에 하기 위한 것입니다;
- MqlTradeResult에 거래 요청에 대한 OrderSend() 함수로 보낸 거래 서버 응답이 포함되어 있습니다;
- MqlTradeTransaction에 거래에 대한 설명이 포함되어 있습니다;
- MqlTick은 현재 가격에 대해 가장 많이 요청된 정보를 신속하게 검색할 수 있도록 설계되었습니다.
- 이코노믹 캘린더 구조는 MetaTrader 5 플랫폼으로 전송된 이코노믹 캘린더 이벤트에 대한 데이터를 실시간으로 얻는 데 사용됩니다. 이코노믹 캘린더 기능은 새로운 보고서가 발표된 직후 거시 경제적 매개변수를 분석할 수 있는데, 이는 관련 값이 소스로부터 지체없이 직접 브로드캐스트 되기 때문입니다.

© 2000-2025, MetaQuotes Ltd.

MqlDateTime

날짜 유형 구조에는 int 유형의 필드 8개가 포함되어 있습니다:

```
struct MqlDateTime
  {
   int year;            // 연
   int mon;             // 개월
   int day;             // 일
   int hour;            // 시간
   int min;             // 분
   int sec;             // 초
   int day_of_week;     // 요일 (0-일, 1-월, ... ,6-토)
   int day_of_year;     // 해의 날짜 번호 (1월 1일에 0의 숫자 값이 할당됨)
  };
```

참고

3월 이후 윤년의 일 수는 비 윤년의 해당 일 수와 다릅니다.

예제:

```
void OnStart()
  {
//---
   datetime date1=D'2008.03.01';
   datetime date2=D'2009.03.01';

   MqlDateTime str1,str2;
   TimeToStruct(date1,str1);
   TimeToStruct(date2,str2);
   printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,
          str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, day of year = %d",str2.day,str2.mon,
          str2.year,str2.day_of_year);
  }
/*  결과:
   01.03.2008, day of year = 60
   01.03.2009, day of year = 59
*/
```

더 보기

TimeToStruct, 구조 및 클래스

© 2000-2025, MetaQuotes Ltd.

지표 입력 매개변수 구조(MqlParam)

MqlParam 구조는 입력 매개변수를 제공하도록 특별히 설계되었습니다 – 기술 지표 핸들을 IndicatorCreate() 함수를 사용해 만들 때.

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                // 입력 매개 변수의 유형, 값: ENUM_DATATYPE
   long              integer_value;       // 정수 유형을 저장하기 위한 필드
   double            double_value;        // double 유형을 저장하기 위한 필드
   string            string_value;        // 문자열 유형을 저장하기 위한 필드
  };
```

지표의 모든 입력 매개 변수는 MqlParam 유형의 배열 형태로 전송됩니다. 이 배열의 각 요소의 *type* 필드는 요소가 전송하는 데이터 유형을 지정합니다. 지표 값은 각 요소(*integer_value*, *double_value* 또는 *string_value*)에 대해 ENUM_DATATYPE 열거값 –*type* 필드에 지정된 –에 따라 적절한 필드에 먼저 배치해야 합니다.

IND_CUSTOM 값이 IndicatorCreate() 함수에 대한 지표 유형으로 세 번째로 전달되는 경우 입력 매개 변수 배열의 첫 번째 요소에는 *type* 필드 –ENUM_DATATYPE 열거값의 TYPE_STRING 이 포함된 –가 있어야 하며 *string_value* 필드에는 커스텀 지표의 이름이 포함되어 있어야 합니다.

© 2000–2025, MetaQuotes Ltd.

MqlRates

이 구조는 가격, 볼륨 및 스프레드에 대한 정보를 저장합니다.

```
struct MqlRates
  {
   datetime time;         // 구간 시작 시간
   double   open;         // 시가
   double   high;         // 구간 중 최고가
   double   low;          // 구간 중 최저가
   double   close;        // 종가
   long     tick_volume;  // 틱 볼륨
   int      spread;       // 스프레드
   long     real_volume;  // 거래량
  };
```

예제:

```
void OnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("가격 데이터 복제 오류 ",GetLastError());
   else Print("복제됨 ",ArraySize(rates)," bars");
  }
```

더 보기

CopyRates, 시계열 액세스

© 2000-2025, MetaQuotes Ltd.

MqlBookInfo

시장 깊이 데이터에 대한 정보를 제공합니다.

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE    유형;            // 주문 유형 ENUM_BOOK_TYPE 열거값
   double            price;           // 가격
   long              volume;          // 거래량
   double            volume_real;     // 더 정확한 볼륨
  };
```

참고

MqlBookInfo 구조는 미리 정의되어 있으므로 선언 및 설명이 필요하지 않습니다. 구조를 사용하려면 이 유형의 변수를 선언하기만 하면 됩니다.

DOM은 일부 기호에 대해서만 사용할 수 있습니다.

예제:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo 정보 ",Symbol());
     }
   else
     {
      Print("심볼에서 DOM 을 받아오는 데 실패함 ",Symbol());
     }
```

더 보기

MarketBookAdd, MarketBookRelease, MarketBookGet, DOM 거래 주문, 데이터 유형

© 2000-2025, MetaQuotes Ltd.

거래 요청 구조 (MqlTradeRequest)

주문 배치 작업을 실행하기 위한 클라이언트 터미널과 거래 서버 간의 상호 작용은 거래 요청을 사용하여 수행됩니다. 거래 요청은 거래 딜을 수행하는 데 필요한 모든 필드를 포함하는 MqlTradeRequest 유형의 특별한 사전 정의된 구조로 표시됩니다. 요청 처리 결과는 MqlTradeResult 유형의 구조로 표시됩니다.

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;           // 거래 작업 유형
   ulong                         magic;            // 엑스퍼트 어드바이저 ID (매직 넘버)
   ulong                         order;            // 주문 티켓
   string                        symbol;           // 거래 기호
   double                        volume;           // 로트 단위 거래 요청 볼륨
   double                        price;            // 가격
   double                        stoplimit;        // 주문의 StopLimit 레벨
   double                        sl;               // 주문의 손실 수준 중지
   double                        tp;               // 주문의 수익 취득 수준
   ulong                         deviation;        // 요청된 가격에서 최대 가능 편차
   ENUM_ORDER_TYPE               type;             // 주문 유형
   ENUM_ORDER_TYPE_FILLING       type_filling;     // 주문 실행 유형
   ENUM_ORDER_TYPE_TIME          type_time;        // 주문 만료 유형
   datetime                      expiration;       // 주문 만료 시간 (ORDER_TIME_SPECIF
   string                        comment;          // 주문 코멘트
   ulong                         position;         // 포지션 티켓
   ulong                         position_by;      // 반대 포지션의 티켓
  };
```

필드 설명

필드	설명
액션	거래 작업 유형. ENUM_TRADE_REQUEST_ACTIONS 열거 값 중 하나일 수 있습니다.
매직	엑스퍼트 어드바이저 ID. 거래 주문의 분석 처리를 체계화 할 수 있습니다. 각 엑스퍼트 어드바이저는 거래 요청을 보낼 때 고유한 ID를 설정할 수 있습니다.
주문	주문 티켓. 보류 중인 주문을 수정하는 데 사용됩니다.
심볼	주문의 심볼. 주문 수정 및 포지션 닫기 작업에는 필요하지 않습니다.
볼륨	요청된 로트 단위 주문량. 거래의 실제 볼륨은 주문 실행 유형에 따라 달라집니다.
가격	가격, 무문을 실행해야 하는 수준입니다. 실행 유형이 (SYMBOL_TRADE_EXECUTION_MARKET) TRADE_ACTION_DEAL 유형의 "시장 실행"인 기호의 시장 주문에는 가격 사양이 필요하지 않습니다.
stoplimit	가격이 *가격* 가치에 도달했을 때(해당 조건이 의무임) 제한 보류 중인 주문이 배치되는 가격 가치. 그 때까지 보류 중인 주문이 처리되지 않습니다.
sl	가격 이동이 불리한 경우 손실 가격 중지하기
tp	가격 이동이 유리한 경우 이익 가격 취득

© 2000-2025, MetaQuotes Ltd.

필드	설명
편차	포인트로 지정된 최대 가격 편차
유형	주문 유형. ENUM_ORDER_TYPE 열거 값 중 하나일 수 있습니다.
type_filling	주문 실행 유형. ENUM_ORDER_TYPE_FILLING 열거 값 중 하나일 수 있습니다.
type_time	주문 만료 유형. ENUM_ORDER_TYPE_TIME 열거 값 중 하나일 수 있습니다.
만료	주문 만료 시간 (ORDER_TIME_SPECIFIED 유형 주문의 경우)
코멘트	주문 코멘트
포지션	포지션의 티켓. 위치를 식별하기 위해 포지션을 수정하거나 닫을 때 채워야 합니다. 원칙적으로 그것은 그 포지션이 열린 주문의 티켓과 같습니다.
position_by	반대 포지션의 티켓. 위치가 반대 방향으로 동일한 기호에 대해 반대 방향으로 열려 있을 때 사용됩니다.

위험 회피 시스템에서 위치를 수정하거나 닫을 때 해당 티켓(MqlTradeRequest::position)을 지정해야 합니다. 위치는 기호 이름으로 식별되지만 티켓은 네트 시스템에서 지정할 수도 있습니다.

거래 작업을 수행하기 위해 주문을 전송하려면 OrderSend() 기능을 사용해야 합니다. 각 거래 작업에 대해 의무 필드를 지정해야 하며, 옵션 필드도 채워질 수 있습니다. 거래 주문을 보낼 수 있는 7가지 경우가 있습니다:

요청 실행

요청 실행 모두에서 포지션을 열기 위한 거래 주문입니다(요청된 가격에 따라 거래). 다음 9개의 필드를 지정해야 합니다:

- 액션
- 심볼
- 볼륨
- 가격
- sl
- tp
- 편차
- 유형
- type_filling

또한 "매직" 및 "코멘트" 필드 값을 지정할 수 있습니다.

인스턴트 실행

인스턴트 실행 모드에서 포지션을 열기 위한 거래 순서입니다(현재 가격 기준 거래). 다음 9개의 필드를 지정해야 합니다:

- 액션
- 심볼
- 볼륨
- 가격
- sl

© 2000-2025, MetaQuotes Ltd.

- tp
- 편차
- 유형
- type_filling

또한 "매직" 및 "코멘트" 필드 값을 지정할 수 있습니다.

마켓 실행

마켓 실행 모드에서 포지션을 열기 위한 거래 순서입니다. 다음 5개의 필드를 지정해야 합니다:

- 액션
- 심볼
- 볼륨
- 유형
- type_filling

또한 "매직" 및 "코멘트" 필드 값을 지정할 수 있습니다.

교환 실행

교환 실행 모드에서 포지션을 열기 위한 거래 순서입니다. 다음 5개의 필드를 지정해야 합니다:

- 액션
- 심볼
- 볼륨
- 유형
- type_filling

또한 "매직" 및 "코멘트" 필드 값을 지정할 수 있습니다.

매입 포지션을 열기 위한 TRADE_ACTION_DEAL 거래 작업의 예:

```
#define EXPERT_MAGIC 123456   // MagicNumber of the expert
//+------------------------------------------------------------------+
//| 매입 위치 열기                                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
//--- 요청의 매개변수
   request.action   =TRADE_ACTION_DEAL;                     // 거래 작업 유형
   request.symbol   =Symbol();                              // symbol
   request.volume   =0.1;                                   // 0.1 로트의 볼륨
   request.type     =ORDER_TYPE_BUY;                        // 주문 유형
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // 개장 가격
   request.deviation=5;                                     // 가격에 대한 허용 편차
   request.magic    =EXPERT_MAGIC;                          // 주문의 MagicNumber
//--- 요청 전송
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // if unable to send the r
//--- 작업에 대한 정보
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

매도 포지션을 열기 위한 TRADE_ACTION_DEAL 거래 작업의 예:

```
#define EXPERT_MAGIC 123456   // 엑스퍼트의 MagicNumber
//+------------------------------------------------------------------+
//| 매도 위치 열기                                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
//--- 요청의 매개변수
   request.action   =TRADE_ACTION_DEAL;                     // 거래 작업 유형
   request.symbol   =Symbol();                              // symbol
   request.volume   =0.2;                                   // 0.2 로트의 볼륨
   request.type     =ORDER_TYPE_SELL;                       // 주문 유형
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // 개장 가격
   request.deviation=5;                                     // 가격에 대한 허용 편차
   request.magic    =EXPERT_MAGIC;                          // MagicNumber of the orde
//--- 요청 전송
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());     // if unable to send the r
//--- 작업에 대한 정보
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

마감 포지션에 대한 TRADE_ACTION_DEAL 거래 작업의 예:

© 2000-2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456   // 엑스퍼트의 MagicNumber
//+------------------------------------------------------------------+
//| 모든 포지션 닫기                                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청 및 거래 요청 결과 선언하고 초기화
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 열린 포지션의 숫자
//--- 모든 열린 포지션에서 반복
   for(int i=total-1; i>=0; i--)
     {
      //--- 주문의 매개변수
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      double volume=PositionGetDouble(POSITION_VOLUME);
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- 포지션에 대한 출력 정보
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- MagicNumber가 일치하면
      if(magic==EXPERT_MAGIC)
        {
         //--- 요청 및 결과 값 0 설정
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 작업 매개변수 설정
         request.action   =TRADE_ACTION_DEAL;        // 거래 작업의 유형
         request.position =position_ticket;          // t포지션의 티켓
         request.symbol   =position_symbol;          // 기호
         request.volume   =volume;                   // 포지션의 볼륨
         request.deviation=5;                        // 가격에서의 허용 편차
         request.magic    =EXPERT_MAGIC;             // 포지션의 MagicNumber
         //--- 포지션에 따라 가격과 주문의 종류를 설정
         if(type==POSITION_TYPE_BUY)
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_BID);
            request.type =ORDER_TYPE_SELL;
           }
         else
           {
            request.price=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
            request.type =ORDER_TYPE_BUY;
           }
         //--- 마감에 대한 정보 출력
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToString
         //--- 요청 전송
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // 요청을 보낼 수 없는 경
         //--- 작업에 대한 정보
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
```

SL & TP 수정

StopLoss 및/또는 TakeProfit 가격 수준을 수정하기 위한 거래 주문. 다음 4개의 필드를 지정해야 합니다:

- 액션
- 심볼
- sl
- tp
- 포지션

열린 포지션의 손실 중지 및 이익 취하기 값을 수정하기 위한 TRADE_ACTION_SLTP 거래 작업의 예:

```
#define EXPERT_MAGIC 123456  // 엑스퍼트의 MagicNumber
//+------------------------------------------------------------------+
//| 손실 정지 수정 및 포지션 이익 취득                               |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // 열린 포지션의 숫자
//--- 모든 열린 포지션에서 반복
   for(int i=0; i<total; i++)
     {
      //--- 주문의 매개변수
      ulong  position_ticket=PositionGetTicket(i);// 포지션의 티켓
      string position_symbol=PositionGetString(POSITION_SYMBOL); // 기호
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // 소수점 0
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // 포지션의 MagicNumber
      double volume=PositionGetDouble(POSITION_VOLUME);    // 포지션의 볼륨
      double sl=PositionGetDouble(POSITION_SL);  // 포지션의 손실 중지
      double tp=PositionGetDouble(POSITION_TP);  // 포지션의 이익 가져가기
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- 포지션에 대한 정보 출력
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- MagicNumber가 일치하는 경우 손실 중지 및 이익 취득이 정의되지 않습니다
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- 현재 가격 수준 계산
         double price=PositionGetDouble(POSITION_PRICE_OPEN);
         double bid=SymbolInfoDouble(position_symbol,SYMBOL_BID);
         double ask=SymbolInfoDouble(position_symbol,SYMBOL_ASK);
         int    stop_level=(int)SymbolInfoInteger(position_symbol,SYMBOL_TRADE_STOPS_L
         double price_level;
         //--- 현재 마감 가격에서 허용되는 최소 편치 거리(포인트)가 설정되지 않은 경우
         if(stop_level<=0)
            stop_level=150; // 현행 종가 대비 편치 거리를 150 포인트로 설정
         else
            stop_level+=50; // 신뢰성에 대해 편치 거리를 (SYMBOL_TRADE_STOPS_LEVEL + 50)

         //--- 손실 중지 및 이익 취득 값의 계산 및 반올림
         price_level=stop_level*SymbolInfoDouble(position_symbol,SYMBOL_POINT);
         if(type==POSITION_TYPE_BUY)
           {
            sl=NormalizeDouble(bid-price_level,digits);
            tp=NormalizeDouble(bid+price_level,digits);
           }
         else
           {
            sl=NormalizeDouble(ask+price_level,digits);
            tp=NormalizeDouble(ask-price_level,digits);
           }
         //--- 요청 및 결과 값 0 설정
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 작업 매개변수 설정
         request.action  =TRADE_ACTION_SLTP; // 거래 작업의 유형
         request.position=position_ticket;   // 포지션의 티켓
         request.symbol=position_symbol;     // 기호
         request.sl      =sl;                // 포지션의 손실 중지
         request.tp      =tp;                // 포지션의 이익 취득
         request.magic=EXPERT_MAGIC;         // 포지션의 MagicNumber
         //--- 수정 정보 출력
         PrintFormat("Modify #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- 요청 전송
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // 요청을 보낼 수 없는 경
         //--- 작업에 대한 정보
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

보류 중인 주문

보류 중인 주문을 발주하기 위한 거래 주문 다음 11개 필드를 지정해야 합니다:

- 액션
- 심볼
- 볼륨
- 가격
- stoplimit
- sl
- tp
- 유형

© 2000-2025, MetaQuotes Ltd.

- type_filling
- type_time
- 만료

또한 "매직" 및 "코멘트" 필드 값을 지정할 수 있습니다.

보류 중인 주문을 발주하기 위한 TRADE_ACTION_PENDING 거래 작업의 예:

© 2000-2025, MetaQuotes Ltd.

```
#property description "Example of placing pending orders"
#property script_show_inputs
#define EXPERT_MAGIC 123456                                  // 엑스퍼트의 MagicNumber
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // 주문 유형
//+------------------------------------------------------------------+
//| 보류 중인 주문 발주                                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
//--- 보류 중인 주문을 발주할 매개 변수
   request.action   =TRADE_ACTION_PENDING;                                  // 거래 작업 유형
   request.symbol   =Symbol();                                              // 기호
   request.volume   =0.1;                                                   // 0.1 로트의 볼
   request.deviation=2;                                                     // 가격에 대한 허
   request.magic    =EXPERT_MAGIC;                                          // 주문의 Magic
   int offset = 50;                                                         // 주문을 발주할
   double price;                                                            // 가격 유발 주문
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);                     // 포인트의 값
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);                     // 소수 자릿수(숫
   //--- 작업 유형 확인
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                               // order type
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;             // 개장 가격
      request.price    =NormalizeDouble(price,digits);                      // 정규화된 개시
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                               // 주문 유형
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;             // 개장 가격
      request.price    =NormalizeDouble(price,digits);                      // 정규화된 시가
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                                    // 주문 유형
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;     // 개장 가격
      request.price=NormalizeDouble(price,digits);                          // 정규화된 시가
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                               // 주문 유형
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;             // 개장 가격
      request.price    =NormalizeDouble(price,digits);                      // 정규화된 시가
     }
   else Alert("이 예는 보류 중인 주문을 작성하는 경우에만 해당");   // 보류 중인 주문이 선택되
//--- 요청 전송
   if(!OrderSend(request,result))
      PrintFormat("OrderSend error %d",GetLastError());                     // 요청을 보낼
//--- 작업 정보
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

보류 중인 주문 수정

보류 중인 주문의 가격을 수정하기 위한 거래 주문. 다음 7개의 필드를 지정해야 합니다:

- 액션

© 2000-2025, MetaQuotes Ltd.

- 주문
- 가격
- sl
- tp
- type_time
- 만료

예시 - TRADE_ACTION_MODIFY 거래 작업 (보류 중인 주문의 가격 수준을 수정하기 위함):

© 2000–2025, MetaQuotes Ltd.

```
#define EXPERT_MAGIC 123456  // 엑스퍼트의 Magi cNumber
//+------------------------------------------------------------------+
//| 보류 중인 주문 수정                                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래와 거래 요청의 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
   int total=OrdersTotal(); // 보류 중인 총 주문 수
//--- 보류 중인 주문을 전부 반복
   for(int i=0; i<total; i++)
     {
      //--- 주문의 매개변수
      ulong  order_ticket=OrderGetTicket(i);                                  // 주문 티켓
      string order_symbol=Symbol();                                           // 기호
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS);       // 소수점 이하
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                              // 주문의 Magi
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);                     // 주문의 현재
      double sl=OrderGetDouble(ORDER_SL);                                     // current St
      double tp=OrderGetDouble(ORDER_TP);                                     // 현재 주문의
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE);      // 주문의 유형
      int offset = 50;                                                        // 주문을 발주
      double price;                                                           // 가격 유발 ?
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);               // 포인트의 값
      //--- 주문 정보 출력
      PrintFormat("#%I64u %s  %s  %.2f  %s  sl: %s  tp: %s  [%I64d]",
                  order_ticket,
                  order_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  DoubleToString(sl,digits),
                  DoubleToString(tp,digits),
                  magic);
      //--- Magi cNumber가 일치하지 않는 경우 손실 중지 및 이익 취득이 정의되지 않습니다
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                                  // 거래 작업의 유
         request.order = OrderGetTicket(i);                                   // 주문 티켓
         request.symbol   =Symbol();                                          // 기호
         request.deviation=5;                                                 // 가격에 대한 허
        //--- 가격 수준 설정, 주문 유형에 따라 이익 가져가기 및 손실 중지
         if(type==ORDER_TYPE_BUY_LIMIT)
           {
            price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 정규화된
           }
         else if(type==ORDER_TYPE_SELL_LIMIT)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                  // 정규화
           }
         else if(type==ORDER_TYPE_BUY_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;
            request.tp = NormalizeDouble(price+offset*point,digits);
```

© 2000-2025, MetaQuotes Ltd.

```
            request.sl = NormalizeDouble(price-offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                    // 정규화
           }
         else if(type==ORDER_TYPE_SELL_STOP)
           {
           price = SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;
            request.tp = NormalizeDouble(price-offset*point,digits);
            request.sl = NormalizeDouble(price+offset*point,digits);
            request.price    =NormalizeDouble(price,digits);                    // 정규화
           }
         //--- 요청 전송
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // 요청 전송이 불가능한 경
         //--- 작업 정보
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- 요청 및 결과 값 0 설정
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

보류 중인 주문 삭제

보류 중인 주문을 삭제하기 위한 거래 주문. 다음 두 개의 필드를 지정해야 합니다:

- 액션
- 주문

보류 중인 주문을 삭제하기 위한 TRADE_ACTION_REMOVE 거래 작업의 예:

```
#define EXPERT_MAGIC 123456  // 엑스퍼트의 Magi cNumber
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 보류 중인 주문 삭제                                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래 요청과 거래 요청 결과를 선언하고 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
   int total=OrdersTotal(); // 보류 중인 총 주문 수
//--- 보류 중인 주문을 전부 반복
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // 주문 티켓
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // 주문의 MagicNumber
      //--- MagicNumber가 일치하면
      if(magic==EXPERT_MAGIC)
        {
         //--- 요청 및 결과 값 0 설정
         ZeroMemory(request);
         ZeroMemory(result);
         //--- 작업 매개변수 설정
         request.action=TRADE_ACTION_REMOVE;                   // 거래 작업 유형
         request.order = order_ticket;                         // 주문 티켓
         //--- 요청 전송
         if(!OrderSend(request,result))
            PrintFormat("OrderSend error %d",GetLastError());  // 요청을 보낼 수 없는 경
         //--- 작업 정보
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

추가 참조

구조 및 클래스, 거래 함수, 주문 속성

© 2000-2025, MetaQuotes Ltd.

거래 요청 확인 결과 구조 (MqlTradeCheckResult)

거래 서버에 거래 작업 요청을 보내기 전에 먼저 확인하는 것이 좋습니다. 확인은 OrderCheck() 함수를 사용하여 수행되며, 이 함수에 선택된 요청과 MqlTradeCheckResult 구조 유형의 변수가 전달됩니다. 확인 결과가 이 변수에 기록됩니다.

```
struct MqlTradeCheckResult
  {
   uint         retcode;             // 회신 코드
   double       balance;             // 거래 실행 후 잔액
   double       equity;              // 거래 실행 후 지분
   double       profit;              // 플로팅 이익
   double       margin;              // 마진 요구사항
   double       margin_free;         // 여유 마진
   double       margin_level;        // 마진 레벨
   string       comment;             // 응답 코드에 대한 코멘트(오류에 대한 설명)
  };
```

필드에 대한 설명

필드	설명
retcode	반환 코드
balance	거래 작업 실행 후 잔액 값
equity	거래 작업 실행 후 지분 값
profit	거래 작업 실행 후 플로팅 이익의 값
margin	거래 작업에 필요한 마진
margin_free	거래 작업 실행 후 남은 여유 마진
margin_level	거래 작업 실행 이후 설정될 마진 수준
코멘트	응답 코드에 대한 설명, 오류 설명

추가 참조

거래 요청 구조, 현재 가격 구조, OrderSend, OrderCheck

© 2000-2025, MetaQuotes Ltd.

거래 요청 결과 구조 (MqlTradeResult)

거래 서버는 거래 요청의 결과로 거래 요청 처리 결과에 대한 데이터를 MqlTradeResult type 유형의 특수 사전정의된 구조로 반환합니다.

```
struct MqlTradeResult
  {
   uint     retcode;          // 작업 반환 코드
   ulong    deal;             // 수행된 경우 티켓 거래
   ulong    order;            // 발주된 경우 티켓 주문
   double   volume;           // 브로커가 확인한 딜 볼륨
   double   price;            // 브로커가 확인한 딜 가격
   double   bid;              // 현재 입찰 가격
   double   ask;              // 현재 호가
   string   comment;          // 브로커 설명 작업(기본적으로 거래 서버 반환 코드에 대한 설명으
   uint     request_id;       // 발송 중 터미널에서 설정한 요청 ID
   int      retcode_external; // 외부 트레이딩 시스템의 반환 코드
  };
```

필드 설명

필드	설명
retcode	거래 서버의 반환 코드
딜	거래가 성사된 경우 티켓 거래. TRADE_ACTION_DEAL 유형의 거래 작업에 사용할 수 있습니다
주문	발주된 경우 티켓을 주문. TRADE_ACTION_PENDING 유형의 거래 작업에 사용할 수 있습니다
볼륨	브로커가 확인한 딜 볼륨. 주문 채우기 유형에 따라 다릅니다
가격	브로커가 확인한 거래 가격. *편차* 필드 (거래 요청) 및/또는 거래 작업에 따라 다릅니다
입찰	현재 시장 입찰 가격 (재산정 가격)
요구	현재 시장 호가 (재산정 가격)
코멘트	작업에 대한 브로커 설명 (기본적으로 거래 서버 반환 코드에 대한 설명으로 채워짐)
request_id	거래 서버로 보낼 때 터미널에서 설정한 요청 ID
retcode_external	외부 거래 시스템에서 반환한 오류 코드. 이러한 오류의 용도 및 유형은 거래 작업이 전송되는 브로커 및 외부 거래 시스템에 따라 달라집니다.

거래 작업 결과가 MqlTradeResult 유형의 변수로 반환되며, 이 변수는 OrderSend()에 두 번째 매개변수로 거래 작업을 수행하기 위해 전달됩니다.

터미널은 요청 ID를 거래 서버로 전송할 때 OrdersSend() 및 OrderSendAsync() 함수를 사용하여 request_id 필드에서 수정합니다. 터미널은 거래 서버로부터 수행된 거래에 대한 메시지를 수신하고 다음 구성 요소를 매개 변수로 포함하는 OnTradeTransaction() 함수로 처리할 수 있도록 제출합니다:

© 2000-2025, MetaQuotes Ltd.

- MqlTradeTransaction 구조의 거래 설명;
- OrderSend() 또는 OrdersSendAsync() 기능에서 보낸 거래 요청에 대한 설명입니다. 요청 ID 터미널에서 거래 서버로 전송되고 요청 자체와 해당 request_id는 터미널 메모리에 저장됩니다;
- 이 요청의 ID가 포함된 request_id 필드가 있는 MqlTradeResult 구조로 된 거래 요청 실행 결과 있니다.

OnTradeTransaction() 함수는 3개의 입력 매개변수를 수신하지만 마지막 2개는 TRADE_TRANSACTION_REQUEST 유형의 트랜잭션에 대해서만 분석해야 합니다. 그 밖의 모든 경우에는 거래 요청 자료와 그 실행 결과에 대한 자료가 채워지지 않습니다. 매개변수 분석 예는 거래 요청 구조에서 확인할 수 있습니다.

거래 요청을 서버로 보낼 때 단말기가 request_id를 설정하는 것은 주로 OrderSendAsync() 비동기 함수로 작업할 때 도입됩니다. 이 지표를 사용하면 수행된 작업(OrderSend 또는 OrderSendAsync 함수 호출)을 OnTradeTransaction()으로 전송된 이 작업의 결과와 연결할 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 결과 처리와 함께 거래 요청 전송                                  |
//+------------------------------------------------------------------+
bool MyOrderSend(MqlTradeRequest  요청,MqlTradeResult  결과)
  {
//--- 마지막 오류 코드를 0으로 재설정
   ResetLastError();
//--- 요청 전송
   bool success=OrderSend(request,result);
//--- 결과가 실패하는 경우 - 그 이유를 알아보세요
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog: 거래 요청 실패함. Error = ",GetLastError());
      switch(answer)
        {
         //--- 재산정
         case 10004:
           {
            Print("TRADE_RETCODE_REQUOTE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- 서버에서 주문을 수락하지 않습니다
         case 10006:
           {
            Print("TRADE_RETCODE_REJECT");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- 유효하지 않은 가격
         case 10015:
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             Print("TRADE_RETCODE_INVALID_PRICE");
             Print("request.price = ",request.price,"   result.ask = ",
                   result.ask," result.bid = ",result.bid);
             break;
            }
          //--- 유효하지 않은 SL 및/또는 TP
          case 10016:
            {
             Print("TRADE_RETCODE_INVALID_STOPS");
             Print("request.sl = ",request.sl," request.tp = ",request.tp);
             Print("result.ask = ",result.ask," result.bid = ",result.bid);
             break;
            }
          //--- 유효하지 않은 볼륨
          case 10014:
            {
             Print("TRADE_RETCODE_INVALID_VOLUME");
             Print("request.volume = ",request.volume,"   result.volume = ",
                   result.volume);
             break;
            }
          //--- 거래 작업을 위한 충분한 자금이 없습니다
          case 10019:
            {
             Print("TRADE_RETCODE_NO_MONEY");
             Print("request.volume = ",request.volume,"   result.volume = ",
                   result.volume,"   result.comment = ",result.comment);
             break;
            }
          //--- 다른 이유로 서버 응답 코드를 출력
          default:
            {
             Print("Other answer = ",answer);
            }
         }
       //--- 거래 요청의 실패 결과를 false로 반환하여 알림
       return(false);
      }
//--- OrderSend()가 true를 반환 - 답변을 반복
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

매매 거래의 구조 (MqlTradeTransaction)

거래 계좌에 대해 몇 가지 명확한 작업을 수행하면 상태가 변경됩니다. 이러한 작업에는 다음이 포함됩니다:

- OrderSend 및 OrderSendAsync 기능과 추가 실행을 사용하여 클라이언트 터미널의 MQL5 응용 프로그램에서 거래 요청 전송;
- 터미널 그래픽 인터페이스 및 추가 실행을 통해 거래 요청 전송;
- 서버에서 주문 보류 및 주문 활성화 중지;
- 거래 서버측에서 작업 수행.

이러한 조치의 결과로 다음과 같은 매매 거래가 수행됩니다:

- 거래 요청 처리;
- 공개 주문 변경;
- 주문 내역 변경;
- 거래 내역 변경;
- 포지션 변경.

예를 들어, 시장 매수 주문을 보낼 때 처리되고, 계정에 대한 적절한 매수 주문이 생성되며, 주문이 실행되어 열린 주문 목록에서 삭제되고, 주문 기록에 추가되고, 적절한 거래가 내역에 추가되고 새로운 포지션이 생성됩니다. 이 모든 행위는 매매 거래입니다.

특수 OnTradeTransaction() 핸들러는 계정에 적용되는 매매 거래를 가져오기 위해 제공됩니다. 처리기의 첫 번째 매개 변수는 MqlTradeTransaction 구조를 가져와 매매 거래를 설명합니다.

```
struct MqlTradeTransaction
  {
   ulong                     deal;             // 거래 티켓
   ulong                     order;            // 주문 티켓
   string                    symbol;           // 거래 기호 이름
   ENUM_TRADE_TRANSACTION_TYPE type;           // 매매 거래 유형
   ENUM_ORDER_TYPE           order_type;       // 주문 유형
   ENUM_ORDER_STATE          order_state;      // 주문 상태
   ENUM_DEAL_TYPE            deal_type;        // 거래 유형
   ENUM_ORDER_TYPE_TIME      time_type;        // 행위 기간별 주문 유형
   datetime                  time_expiration;  // 주문 만료 시간
   double                    price;            // 가격
   double                    price_trigger;    // 주문 활성화 가격 제한 중지
   double                    price_sl;         // 손실 중지 수준
   double                    price_tp;         // 이익 취득 수준
   double                    volume;           // 볼륨(로트)
   ulong                     position;         // 포지션 티켓
   ulong                     position_by;      // 반대 포지션의 티켓
  };
```

필드 설명

© 2000-2025, MetaQuotes Ltd.

필드	설명
딜	거래 티켓.
주문	주문 티켓.
심볼	트랜잭션이 수행되는 트레이딩 기호 이름.
유형	매매 거래 유형. 값은 ENUM_TRADE_TRANSACTION_TYPE 열거 값 중 하나일 수 있습니다.
order_type	거래 주문 유형. 값은 ENUM_ORDER_TYPE 열거 값 중 하나일 수 있습니다.
order_state	거래 주문 상태. 값은 ENUM_ORDER_STATE 열거 값 중 하나일 수 있습니다.
deal_type	거래 유형. 값은 ENUM_DEAL_TYPE 열거 값 중 하나일 수 있습니다.
type_time	만료 시 주문 유형. 값은 ENUM_ORDER_TYPE_TIME 값 중 하나일 수 있습니다.
time_expiration	주문 만료 기간 보류 중 (ORDER_TIME_SPECIFIED 및 ORDER_TIME_SPECIFIED_DAY 유형 주문의 경우).
가격	가격. 매매 거래 유형에 따라 주문, 거래 또는 포지션의 가격이 될 수 있습니다.
price_trigger	주문 중지 (활성화) 가격 한도 중지 (ORDER_TYPE_BUY_STOP_LIMIT 및 ORDER_TYPE_SELL_STOP_LIMIT).
price_sl	손실 중지 가격. 매매 거래 유형에 따라 주문, 거래 또는 포지션과 관련될 수 있습니다.
price_tp	이익 취득 가격. 매매 거래 유형에 따라 주문, 거래 또는 포지션과 관련될 수 있습니다.
볼륨	볼륨(로트). 매매 거래 유형에 따라 주문, 거래 또는 포지션의 현재 수량을 나타낼 수 있습니다.
포지션	트랜잭션의 영향을 받는 포지션의 티켓.
position_by	반대편 포지션의 티켓. 반대 방향, 즉 반대 방향으로 열린 동일한 기호 포지션에 의해 포지션을 닫을 때 사용됩니다.

수신된 트랜잭션 분석을 위한 필수 매개변수는 **유형** 필드에 지정된 유형입니다. 예를 들어, 트랜잭션이 TRADE_TRANSACTION_REQUEST 유형(서버에서 거래 요청을 처리한 결과)인 경우 구조에는 완전히 채워진 필드 하나인 **유형**만 있습니다. 다른 필드는 분석되지 않습니다. 이 경우, 아래와 같이 OnTradeTransaction() 핸들러에 제출된 두가지 추가 **요청** 및 **결과** 매개변수를 분석할 수 있습니다.

거래 작업 유형에 대한 데이터가 있으면 거래 계정의 주문, 포지션 및 거래 현황 분석을 결정할 수 있습니다. 터미널에서 서버로 전송된 거래 요청 하나로 여러 개의 새 트랜잭션이 생성될 수 있습니다. 터미널에 도착하는 우선순위는 보장되지 않습니다.

MqlTradeTransaction 구조는 매매 거래 유형 (ENUM_TRADE_TRANSACTION_TYPE)에 따라 다양한 방식으로 채워집니다:

TRADE_TRANSACTION_ORDER_* 및 TRADE_TRANSACTION_HISTORY_*

MqlTradeTransaction 구조의 다음 필드는 보류 중인 주문 처리(TRADE_TRANSACTION_ORDER_ADD, TRADE_TRANSACTION_ORDER_UPDATE and TRADE_TRANSACTION_ORDER_DELETE) 및 주문 내역

© 2000-2025, MetaQuotes Ltd.

(TRADE_TRANSACTION_HISTORY_ADD, TRADE_TRANSACTION_HISTORY_UPDATE, TRADE_TRANSACTION_HISTORY_DELETE)과 관련된 매매 거래에 대해 채워집니다:

- order – 주문 티켓;
- symbol – 주문 기호 이름;
- type – 매매 거래 유형;
- order_type – 주문 유형;
- orders_state – 주문 현재 상태;
- time_type – 주문 만료 유형;
- time_expiration – 주문 만료 시간 (ORDER_TIME_SPECIFIED 및 ORDER_TIME_SPECIFIED_DAY 만료 유형이 있는 주문의 경우);
- price – 클라이언트가 지정한 주문 가격;
- price_trigger – 중제 제한 주문 중지 가격 (ORDER_TYPE_BUY_STOP_LIMIT 및 ORDER_TYPE_SELL_STOP_LIMIT인 경우에만);
- price_sl – 손실 중지 주문 가격 (주문에서 지정된 경우 채워짐);
- price_tp – 이익 취득 주문 가격 (주문에서 지정된 경우 채워짐);
- volume – 주문 현재 볼륨(채워지지 않음). 초기 주문량은 HistoryOrders* 기능을 사용하여 주문 내역에서 확인할 수 있습니다.
- position – 주문 실행의 결과로 열렸거나, 수정되었거나, 닫혔던 위치의 티켓입니다. 시장 주문에 대해서만 작성되며 TRADE_TRANSACTION_ORDER_ADD에는 작성되지 않습니다.
- position_by – 반대 포지션의 티켓. 명령에 의해서만 채워집니다(반대 포지션에 의해 닫힘).

TRADE_TRANSACTION_DEAL_*

MqlTradeTransaction 구조의 다음 필드는 거래 처리(TRADE_TRANSACTION_DEAL_ADD, TRADE_TRANSACTION_DEAL_UPDATE 및 TRADE_TRANSACTION_DEAL_DELETE)와 관련된 매매 거래에 대해 채워집니다:

- deal – 거래 티켓;
- order – 거래가 성사된 티켓 주문;
- symbol – 거래 기호 이름;
- type – 매매 거래 유형;
- deal_type – 거래 유형;
- price – 거래 가격;
- price_sl – 손실 중지 가격 (주문에 지정된 경우, 거래가 수행된 기준으로 작성됨);
- price_tp – 이익 취득 가져오기 (주문에 지정된 경우 수행된 기준으로 작성됨);
- volume – 거래 볼륨(로트).
- position – 거래 실행의 결과로 개설, 수정 또는 마감된 포지션의 티켓.
- position_by – 반대 포지션의 티켓. 거래에 의해서만 아웃이 채워집니다(상대방의 포지션을 닫음).

TRADE_TRANSACTION_POSITION

MqlTradeTransaction 구조의 다음 필드는 거래 실행과 연결되지 않은 위치 변경과 관련된 매매 거래에 대해 채워집니다(TRADE_TRANSACTION_POSITION):

- symbol – 포지션 기호 이름;
- type – 매매 거래 유형;
- deal_type – 포지션 유형 (DEAL_TYPE_BUY 또는 DEAL_TYPE_SELL);
- price – 가중 평균 포지션 공개 가격;
- price_sl – 손실 중지 가격;
- price_tp – 이익 취득 가격;
- volume – 볼륨이 변경된 경우 볼륨(로트) 포지션.

© 2000-2025, MetaQuotes Ltd.

> 거래 실행으로 인한 위치 변경(추가, 변경 또는 마감)으로 인해 TRADE_TRANSACTION_POSITION 트랜잭션이 발생하지 않습니다.

TRADE_TRANSACTION_REQUEST

거래 요청이 서버에서 처리되어 처리 결과가 수신되었음을 설명하는 매매 거래에 대해 MqlTradeTransaction 구조이 필드 하나만 채워집니다(TRADE_TRANSACTION_REQUEST):

- type – 매매 거래 유형;

> 해당 트랜잭션에 대해 유형 필드(매매 거래 유형)만 분석해야 합니다. 추가 데이터에 대해 OnTradeTransaction 함수의 두 번째 및 세 번째 매개변수(요청 및 결과)를 분석해야 합니다.

예:

```
input int MagicNumber=1234567;

//--- CTrade 트레이딩 클래스를 사용하도록 설정하고 이 클래스의 변수를 선언
#include <Trade\Trade.mqh>
CTrade 거래;
//--- 보류 중인 주문 설치 및 삭제 플래그
bool pending_done=false;
bool pending_deleted=false;
//--- 보류 중인 주문 티켓이 여기에 저장됩니다
ulong order_ticket;
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 기능                                             |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- MagicNumber를 설정하여 모든 주문을 표시
   trade.SetExpertMagicNumber(MagicNumber);
//--- 거래 요청은 OrderSendAsync() 함수를 사용하여 비동기 모드로 전송됩니다
   trade.SetAsyncMode(true);
//--- 변수를 0으로 초기화
   order_ticket=0;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 기능                                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---보류 중인 주문 설치
   if(!pending_done)
     {
      double ask=SymbolInfoDouble(_Symbol,SYMBOL_ASK);
      double buy_stop_price=NormalizeDouble(ask+1000*_Point,(int)SymbolInfoInteger(_Sy
      bool res=trade.BuyStop(0.1,buy_stop_price,_Symbol);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- BuyStop() 함수가 성공적으로 수행된 경우
      if(res)
        {
         pending_done=true;
         //--- ctrade에서 요청의 결과를 수신합니다
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //--- 보낸 요청에 대한 request_id 가져오기
         uint request_id=trade_result.request_id;
         Print("보류 중인 주문을 설정하기 위한 요청이 전송되었습니다. Request_ID=",request_i
         //--- 주문 티켓 저장 (CTrade로 보내는 비동기 모드를 사용하는 경우 0이 됨)
         order_ticket=trade_result.order;
         //--- 모두 완료됨, OnTick() 핸들러에서 조기 종료
         return;
        }
     }
//--- 보류 중인 주문 삭제
   if(!pending_deleted)
      //--- 추가 확인
      if(pending_done && (order_ticket!=0))
        {
         //--- 보류 중인 주문 삭제 시도
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- 삭제 요청이 성공적으로 전송된 경우
         if(res)
           {
            pending_deleted=true;
            //--- 요청 실행 결과를 가져오기
            MqlTradeResult trade_result;
            trade.Result(trade_result);
            //--- 결과에서 요청 ID를 가져오기
            uint request_id=trade_result.request_id;
            //--- 저널에 표시
            Print("보류 중인 주문 # 삭제 요청이 전송되었습니다",order_ticket,
                  ". Request_ID=",request_id,
                  "\r\n");
            //--- 요청 결과에서 주문 티켓을 고치기
            order_ticket=trade_result.order;
           }
        }
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction 기능                                            |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 열거 값으로 트랜잭션 유형 가져오기
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- 트랜잭션이 요청 처리 결과인 경우 트랜잭션 이름만 표시됩니다
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- 처리된 요청 문자열 이름 표시
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- 요청 결과 설명 표시
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- 삭제 주문 티켓을 OnTick()의 다음 처리 시 보관
      if(result.order!=0)
        {
         //--- 다음 OnTick() 호출 시 이 주문을 티켓으로 삭제
         order_ticket=result.order;
         Print(" Pending order ticket ",order_ticket,"\r\n");
        }
     }
   else // 다른 유형의 트랜잭션에 대한 전체 설명 표시
//--- 수신된 트랜잭션에 대한 설명을 저널에 표시
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------+
//| 트랜잭션 텍스트 설명을 반환                                      |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans)
  {
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Symbol: "+trans.symbol+"\r\n";
   desc+="Deal ticket: "+(string)trans.deal+"\r\n";
   desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
   desc+="Order ticket: "+(string)trans.order+"\r\n";
   desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
   desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
   desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
   desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
   desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
   desc+="Position: "+(string)trans.position+"\r\n";
   desc+="Position by: "+(string)trans.position_by+"\r\n";
//--- 얻은 문자열을 반환
   return desc;
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 거래 요청 텍스트 설명을 반환                                     |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Symbol: "+request.symbol+"\r\n";
   desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Order ticket: "+(string)request.order+"\r\n";
   desc+="Order type: "+EnumToString(request.type)+"\r\n";
   desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
   desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
   desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
   desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
   desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
   desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
   desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
   desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
   desc+="Comment: "+request.comment+"\r\n";
//--- 얻은 문자열을 반환
   return desc;
  }
//+------------------------------------------------------------------+
//| 요청 처리 결과의 텍스트 설명을 반환   |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result)
  {
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Order ticket: "+(string)result.order+"\r\n";
   desc+="Deal ticket: "+(string)result.deal+"\r\n";
   desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
   desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
   desc+="Comment: "+result.comment+"\r\n";
//--- 얻은 문자열을 반환
   return desc;
  }
```

추가 참조

매매 거래 유형, OnTradeTransaction()

© 2000-2025, MetaQuotes Ltd.

현재 가격 반환을 위한 구조 (MqlTick)

이는 심볼의 최신 가격을 저장하기 위한 구조입니다. 현재 가격에 대해 가장 많이 요청되는 정보를 빠르게 검색할 수 있도록 고안되었습니다.

```
struct MqlTick
  {
   datetime     time;          // 마지막 가격 업데이트 시간
   double       bid;           // 현재 입찰 가격
   double       ask;           // 현재 호가
   double       last;          // 마지막 거래 가격 (마지막)
   ulong        volume;        // 현재 마지막 가격의 볼륨
   long         time_msc;      // 가격 마지막 업데이트 시간(밀리초)
   uint         flags;         // 틱 플래그
   double       volume_real;   // 현재 마지막 가격의 볼륨(더 높은 정확도)
  };
```

MqlTick 유형의 변수를 사용하면 SymbolInfoTick() 함수의 단일 호출 내에서 요구, 입찰, 마지막 가격 및 볼륨의 값을 얻을 수 있습니다.

이전 틱에 비해 변경 사항이 있는지 여부에 관계없이 각 틱에 대한 매개변수가 채워집니다. 따라서 틱 내역에서 이전 값을 검색할 필요 없이 과거 어느 순간에나 정확한 가격을 확인할 수 있습니다. 예를 들어, 틱 도착 중에 입찰 가격만 변경되더라도 구조에는 이전 호가, 볼륨 등 다른 매개변수도 포함됩니다.

틱 플래그를 분석하여 정확히 변경된 데이터를 확인할 수 있습니다:

- TICK_FLAG_BID - 입찰 가격이 변경된 틱
- TICK_FLAG_ASK - 호가가 변경된 틱
- TICK_FLAG_LAST - 마지막 거래 가격이 변경된 틱
- TICK_FLAG_VOLUME - 볼륨이 변경된 틱
- TICK_FLAG_BUY - 매수 거래의 결과로 나타난 틱
- TICK_FLAG_SELL - 매도 거래의 결과로 나타난 틱

예:

```
void OnTick()
  {
   MqlTick last_tick;
//---
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      Print(last_tick.time,": Bid = ",last_tick.bid,
            " Ask = ",last_tick.ask,"  Volume = ",last_tick.volume);
     }
   else Print("SymbolInfoTick() failed, error = ",GetLastError());
//---
  }
```

추가 참조

구조 및 클래스, CopyTicks(), SymbolInfoTick()

© 2000-2025, MetaQuotes Ltd.

경제 달력 구조

이 절에서는 MetaTrader 플랫폼에서 직접 사용할 수 있는 경제 달력으로 작업하기 위한 구조를 설명합니다. 경제 달력은 만들어져있는 백과 사전으로 거시 경제 지표, 발표 날짜 및 중요도에 대한 설명이 포함되어 있습니다. 거시 경제 지표의 관련 값은 게시 즉시 MetaTrader 플랫폼으로 전송되며 필요한 지표의 국가, 통화 및 중요도를 시각적으로 추적할 수 있는 태그로 차트에 표시됩니다.

경제 달력 함수를 사용하면 필요한 국가/통화 쌍의 관점에서 사용자 지정 중요도 기준에 따라 들어오는 이벤트의 자동 분석을 수행할 수 있습니다.

국가 설명은 MqlCalendarCountry 구조로 설정됩니다. 이는 CalendarCountryById() 및 CalendarCountries() 함수에서 사용됩니다

```
struct MqlCalendarCountry
  {
   ulong                             id;                // 국가 ID (ISO 3166-1)
   string                            name;              // 국가 텍스트 이름(현재 E
   string                            code;              // 국가 코드명 (ISO 3166-
   string                            currency;          // 국가 통화 코드
   string                            currency_symbol;   // 국가 화폐 심볼
   string                            url_name;          // mql5.com 웹 사이트 URL
  };
```

이벤트 설명은 MqlCalendarEvent 구조로 설정됩니다. 이는 CalendarEventById(), CalendarEventByCountry() 및 CalendarEventByCurrency() 함수에 사용됩니다

```
struct MqlCalendarEvent
  {
   ulong                             id;                // 이벤트 ID
   ENUM_CALENDAR_EVENT_TYPE          type;              // ENUM_CALENDAR_EVENT_T
   ENUM_CALENDAR_EVENT_SECTOR        sector;            // 이벤트와 관련된 섹터
   ENUM_CALENDAR_EVENT_FREQUENCY     frequency;         // 이벤트 빈도
   ENUM_CALENDAR_EVENT_TIMEMODE      time_mode;         // 이벤트 타임 모드
   ulong                             country_id;        // 국가 ID
   ENUM_CALENDAR_EVENT_UNIT          unit;              // 경제 지표 값의 측정 단위
   ENUM_CALENDAR_EVENT_IMPORTANCE    importance;        // 이벤트 중요도
   ENUM_CALENDAR_EVENT_MULTIPLIER    multiplier;        // 경제 지표 값 곱하기
   uint                              digits;            // 소수점 이하 자릿수
   string                            source_url;        // 이벤트가 게시된 소스의 U
   string                            event_code;        // 이벤트 코드
   string                            name;              // 터미널 언어로 된 이벤트
  };
```

이벤트 값은 MqlCalendarValue 구조로 설정됩니다. 이는 CalendarValueById(), CalendarValueHistoryByEvent(), CalendarValueHistory(), CalendarValueLastByEvent() 및 CalendarValueLast() 함수에 사용됩니다

© 2000-2025, MetaQuotes Ltd.

```
struct MqlCalendarValue
  {
   ulong                             id;                // 밸류 ID
   ulong                             event_id;          // 이벤트 ID
   datetime                          time;              // 이벤트 날짜와 시간
   datetime                          period;            // 이벤트 리포팅 피리어드
   int                               revision;          // 리포팅 피리어드와 관련된
   long                              actual_value;      // value가 정해지지 않을 경
   long                              prev_value;        // value가 정해지지 않을 경
   long                              revised_prev_value; // value가 정해지지 않을 경
   long                              forecast_value;    // value가 정해지지 않을 경
   ENUM_CALENDAR_EVENT_IMPACT        impact_type;       // 환율에 대한 가능한 영향
  //--- value를 체크하는 함수
   bool                           HasActualValue(void) const; // actual_value가 있으면
   bool                           HasPreviousValue(void) const; // prev_value가 있으면 tr
   bool                           HasRevisedValue(void) const; // revised_prev_value가
   bool                           HasForecastValue(void) const; // forecast_value가 있으
  //--- valuesfmf 받는 함수
   double                         GetActualValue(void) const; // value가 없으면 actual_
   double                         GetPreviousValue(void) const; // value가 없으면 prev_va
   double                         GetRevisedValue(void) const; // value가 없으면 revised
   double                         GetForecastValue(void) const; // value가 없으면 returns
  };
```

MqlCalendarValue structure는 *actual_value*, *forecast_value*, *prev_value* 와 *revised_prev_value* 필드의 value를 체크하고 세팅하는 메서드를 제공합니다. 만약 아무런 value도 없을 경우 필드는 LONG_MIN (-9223372036854775808)을 저장합니다.

이들 필드에 저장된 value에는 백만이 곱해집니다. 다음의 함수를 사용하여 MqlCalendarValu에서 밸류를 받을때 CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent and CalendarValueLast, 필드 밸류가 LONG_MIN과 같은지 확인해야 합니다; 만약 필드에 밸류가 지정된 경우 밸류를 얻기 위해서는 밸류를 1,000,000로 나누어 줘야 합니다. value를 가져오는 또 다른 방법은 MqlCalendarValue 구조의 함수를 사용하여 value를 확인하고 가져오는 것입니다 .

캘린더 이벤트 처리의 예:

```
//--- 정수 대신 실제 값으로 캘린더 이벤트를 저장하는 구조를 만듭니다.
struct AdjustedCalendarValue
  {
   ulong                             id;                // 밸류 ID
   ulong                             event_id;          // 이벤트 ID
   datetime                          time;              // 이벤트 날짜와 시간
   datetime                          period;            // 이벤트 리포팅 피리어드
   int                               revision;          // 리포팅 피리어드와 관련된
   double                            actual_value;      // 실제 밸류값
   double                            prev_value;        // 이전 밸류값
   double                            revised_prev_value; // 수정된 이전 밸류값
   double                            forecast_value;    // 예상 밸류값
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_CALENDAR_EVENT_IMPACT          impact_type;          // 환율에 대한 잠제적 위험
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
//--- EU 컨트리 코드(ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- 모든 EU관련 이벤트 값을 받음
   MqlCalendarValue values[];
//--- 이벤트를 가져오는 간격을 설정합니다.
   datetime date_from=D'01.01.2021';  // 2021부터 모든 이벤트를 가져옴
   datetime date_to=0;                // 0은 모든 알려진 이벤트와 아직 일어나지 않은 이벤트
//---2021년부터의 EU 관련 이벤트 히스토리를 요청
   if(!CalendarValueHistory(values, date_from, date_to, EU_code))
     {
      PrintFormat("Error! Failed to get events for country_code=%s", EU_code);
      PrintFormat("Error code: %d", GetLastError());
      return;
     }
   else
      PrintFormat("Received event values for country_code=%s: %d",
                  EU_code, ArraySize(values));
//--- 저널에 출력할 배열의 크기를 줄임.
   if(ArraySize(values)>5)
      ArrayResize(values, 5);
//--- 이벤트 값을 실제 값으로 확인하거나 변환하지 않고 그대로 저널에 출력
   Print("Output calendar values as they are");
   ArrayPrint(values);

//--- 필드 값을 체크하고 실제 값으로 변환
//--- 값을 체크하고 얻기 위한 옵션 1
   AdjustedCalendarValue values_adjusted_1[];
   int total=ArraySize(values);
   ArrayResize(values_adjusted_1, total);
//--- 확인하고 조정한 값을 복사
   for(int i=0; i<total; i++)
     {
      values_adjusted_1[i].id=values[i].id;
      values_adjusted_1[i].event_id=values[i].event_id;
      values_adjusted_1[i].time=values[i].time;
      values_adjusted_1[i].period=values[i].period;
      values_adjusted_1[i].revision=values[i].revision;
      values_adjusted_1[i].impact_type=values[i].impact_type;
      //--- 값을 확인하고 1,000,000으로 나눔
      if(values[i].actual_value==LONG_MIN)
         values_adjusted_1[i].actual_value=double("nan");
```

© 2000-2025, MetaQuotes Ltd.

```
      else
         values_adjusted_1[i].actual_value=values[i].actual_value/1000000.;

      if(values[i].prev_value==LONG_MIN)
         values_adjusted_1[i].prev_value=double("nan");
      else
         values_adjusted_1[i].prev_value=values[i].prev_value/1000000.;

      if(values[i].revised_prev_value==LONG_MIN)
         values_adjusted_1[i].revised_prev_value=double("nan");
      else
         values_adjusted_1[i].revised_prev_value=values[i].revised_prev_value/1000000.

      if(values[i].forecast_value==LONG_MIN)
         values_adjusted_1[i].forecast_value=double("nan");
      else
         values_adjusted_1[i].forecast_value=values[i].forecast_value/1000000.;
     }
   Print("The first method to check and get calendar values");
   ArrayPrint(values_adjusted_1);

//--- option 2 to check and get the values
   AdjustedCalendarValue values_adjusted_2[];
   ArrayResize(values_adjusted_2, total);
//--- 확인하고 조정한 값을 복사
   for(int i=0; i<total; i++)
     {
      values_adjusted_2[i].id=values[i].id;
      values_adjusted_2[i].event_id=values[i].event_id;
      values_adjusted_2[i].time=values[i].time;
      values_adjusted_2[i].period=values[i].period;
      values_adjusted_2[i].revision=values[i].revision;
      values_adjusted_2[i].impact_type=values[i].impact_type;
      //--- 확인하고 값을 얻음
      if(values[i].HasActualValue())
         values_adjusted_2[i].actual_value=values[i].GetActualValue();
      else
         values_adjusted_2[i].actual_value=double("nan");

      if(values[i].HasPreviousValue())
         values_adjusted_2[i].prev_value=values[i].GetPreviousValue();
      else
         values_adjusted_2[i].prev_value=double("nan");

      if(values[i].HasRevisedValue())
         values_adjusted_2[i].revised_prev_value=values[i].GetRevisedValue();
      else
         values_adjusted_2[i].revised_prev_value=double("nan");
```

© 2000-2025, MetaQuotes Ltd.

```
      if(values[i].HasForecastValue())
         values_adjusted_2[i].forecast_value=values[i].GetForecastValue();
      else
         values_adjusted_2[i].forecast_value=double("nan");
     }
   Print("The second method to check and get calendar values");
   ArrayPrint(values_adjusted_2);

//--- 값을 얻는 옵션 3 - 확인하지 않음
   AdjustedCalendarValue values_adjusted_3[];
   ArrayResize(values_adjusted_3, total);
//--- 확인하고 조정한 값을 복사
   for(int i=0; i<total; i++)
     {
      values_adjusted_3[i].id=values[i].id;
      values_adjusted_3[i].event_id=values[i].event_id;
      values_adjusted_3[i].time=values[i].time;
      values_adjusted_3[i].period=values[i].period;
      values_adjusted_3[i].revision=values[i].revision;
      values_adjusted_3[i].impact_type=values[i].impact_type;
      //--- 확인하지 않고 값을 얻음
      values_adjusted_3[i].actual_value=values[i].GetActualValue();
      values_adjusted_3[i].prev_value=values[i].GetPreviousValue();
      values_adjusted_3[i].revised_prev_value=values[i].GetRevisedValue();
      values_adjusted_3[i].forecast_value=values[i].GetForecastValue();
     }
   Print("The third method to get calendar values - without checks");
   ArrayPrint(values_adjusted_3);
  }
/*
 다음의 이벤트 값을 얻었습니다 country_code=EU: 1051
  캘린더 값을 있는 그대로 출력
         [id] [event_id]              [time]            [period] [revision]        [act
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0 -922337203
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
  캘린더 값을 얻기 위한 첫번째 메서드
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
  캘린더 값을 덕기 위한 두번째 메서드
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
```

© 2000-2025, MetaQuotes Ltd.

```
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
  캘린더 값을 얻기 위한 세번째 메서드 - 확인하지 않음
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
*/
```

이벤트 빈도는 MqlCalendarEvent 구조에 명시되어 있습니다. 가능한 값은 리스트에서 설정됩니다 **ENUM_CALENDAR_EVENT_FREQUENCY**

ID	설명
CALENDAR_FREQUENCY_NONE	출시 빈도가 설정되지 않았습니다
CALENDAR_FREQUENCY_WEEK	한 주에 한 번 출시됩니다
CALENDAR_FREQUENCY_MONTH	한 달에 한 번 출시됩니다
CALENDAR_FREQUENCY_QUARTER	한 분기에 한 번 출시됩니다
CALENDAR_FREQUENCY_YEAR	일 년에 한 번 출시됩니다
CALENDAR_FREQUENCY_DAY	하루에 한 번 출시됩니다

이벤트 유형은 MqlCalendarEvent 구조에 지정되어 있습니다. 가능한 값은 리스트에서 설정됩니다 **ENUM_CALENDAR_EVENT_TYPE**

ID	설명
CALENDAR_TYPE_EVENT	이벤트 (미팅, 스피치 및 기타)
CALENDAR_TYPE_INDICATOR	지표
CALENDAR_TYPE_HOLIDAY	휴일

이벤트와 관련된 경제 부문은 MqlCalendarEvent 구조에 명시되어 있습니다. 가능한 값은 리스트에서 설정됩니다 **ENUM_CALENDAR_EVENT_SECTOR**

ID	설명
CALENDAR_SECTOR_NONE	섹터가 설정되지 않았습니다
CALENDAR_SECTOR_MARKET	시장, 거래소
CALENDAR_SECTOR_GDP	국내총생산(GDP)

© 2000-2025, MetaQuotes Ltd.

ID	설명
CALENDAR_SECTOR_JOBS	노동시장
CALENDAR_SECTOR_PRICES	가격
CALENDAR_SECTOR_MONEY	자금
CALENDAR_SECTOR_TRADE	거래
CALENDAR_SECTOR_GOVERNMENT	정부
CALENDAR_SECTOR_BUSINESS	비즈니스
CALENDAR_SECTOR_CONSUMER	소비
CALENDAR_SECTOR_HOUSING	주거
CALENDAR_SECTOR_TAXES	세금
CALENDAR_SECTOR_HOLIDAYS	휴일

이벤트 중요도는 MqlCalendarEvent 구조에 명시되어 있습니다. 가능한 값은 리스트에서 설정됩니다 **ENUM_CALENDAR_EVENT_IMPORTANCE**

ID	설명
CALENDAR_IMPORTANCE_NONE	중요도가 설정되지 않았습니다
CALENDAR_IMPORTANCE_LOW	낮은 중요도
CALENDAR_IMPORTANCE_MODERATE	중간 중요도
CALENDAR_IMPORTANCE_HIGH	높은 중요도

이벤트 값을 표시하는 데 사용되는 측정 단위 유형이 MqlCalendarEvent 구조에 명시되어 있습니다. 가능한 값은 리스트에서 설정됩니다 **ENUM_CALENDAR_EVENT_UNIT**

ID	설명
CALENDAR_UNIT_NONE	측정 단위가 설정되지 않았습니다
CALENDAR_UNIT_PERCENT	백분율
CALENDAR_UNIT_CURRENCY	국가 통화
CALENDAR_UNIT_HOUR	시간
CALENDAR_UNIT_JOB	직업
CALENDAR_UNIT_RIG	시추기
CALENDAR_UNIT_USD	USD
CALENDAR_UNIT_PEOPLE	사람

© 2000-2025, MetaQuotes Ltd.

ID	설명
CALENDAR_UNIT_MORTGAGE	담보대출
CALENDAR_UNIT_VOTE	투표
CALENDAR_UNIT_BARREL	배럴
CALENDAR_UNIT_CUBICFEET	입방피트
CALENDAR_UNIT_POSITION	비상업적 순포지션
CALENDAR_UNIT_BUILDING	건물

경우에 따라 경제적 매개변수 값은 MqlCalendarEvent 구조에 승수 집합을 필요로 합니다. 가능한 승수 값이 리스트에서 설정됩니다 **ENUM_CALENDAR_EVENT_MULTIPLIER**

ID	설명
CALENDAR_MULTIPLIER_NONE	승수가 설정되지 않았습니다
CALENDAR_MULTIPLIER_THOUSANDS	수천
CALENDAR_MULTIPLIER_MILLIONS	수백
CALENDAR_MULTIPLIER_BILLIONS	수십억
CALENDAR_MULTIPLIER_TRILLIONS	수조

이벤트가 국가 환율에 미치는 잠재적 영향은 MqlCalendarValue 구조에 나타나 있습니다. 가능한 값은 리스트에서 설정됩니다 **ENUM_CALENDAR_EVENT_IMPACT**

ID	설명
CALENDAR_IMPACT_NA	임팩트가 설정되지 않았습니다
CALENDAR_IMPACT_POSITIVE	긍정 임팩트
CALENDAR_IMPACT_NEGATIVE	부정 임팩트

이벤트 시간은 MqlCalendarEvent 구조에 지정되어 있습니다. 가능한 값은 리스트에서 설정됩니다 **ENUM_CALENDAR_EVENT_TIMEMODE**

ID	설명
CALENDAR_TIMEMODE_DATETIME	소스가 정확한 이벤트 시간을 게시합니다
CALENDAR_TIMEMODE_DATE	이벤트가 하루 종일 소요됨
CALENDAR_TIMEMODE_NOTIME	소스가 이벤트의 시간을 게시하지 않음
CALENDAR_TIMEMODE_TENTATIVE	소스는 정확한 시간이 아닌 이벤트의 날짜를 게시합니다. 시간은 이벤트 발생에 따라 지정됩니다.

© 2000-2025, MetaQuotes Ltd.

더 보기

경제 달력

© 2000-2025, MetaQuotes Ltd.

오류 및 경고 코드

이 절에는 다음과 같은 설명이 포함되어 있습니다:

- 거래 서버의 반환 코드 - 거래 요청 (OrderSend()함수가 전송) 결과 분석;
- 컴파일러 경고 – 컴파일 시 표시되는 경고 메시지 코드(오류 아님);
- 컴파일 오류 – 컴파일 시도 실패 시 오류 메시지 코드;
- 런타임 오류 – mql5 프로그램 실행의 오류 코드 입니다. 오류 코드는 GetLastError() 함수를 사용하여 얻을 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

거래 서버의 반환 코드

거래 작업을 실행하기 위한 거래 요청 MqlTradeRequest의 구조로 전송되는 것은 OrderSend()함수를 사용합니다. 함수 실행 결과는 MqlTradeResult구조에 배치되며 *retcode* 필드에 트레이드 서버 반환 코드가 포함됩니다.

코드	정수	설명
10004	TRADE_RETCODE_REQUOTE	재산정
10006	TRADE_RETCODE_REJECT	요청 거부됨
10007	TRADE_RETCODE_CANCEL	트레이더에 의한 요청 취소
10008	TRADE_RETCODE_PLACED	주문 발주
10009	TRADE_RETCODE_DONE	요청 완료
10010	TRADE_RETCODE_DONE_PARTIAL	요청의 일부분만 완료됨
10011	TRADE_RETCODE_ERROR	요청 진행 오류
10012	TRADE_RETCODE_TIMEOUT	시간 종료에 의한 요청 취소
10013	TRADE_RETCODE_INVALID	유효하지 않은 요청
10014	TRADE_RETCODE_INVALID_VOLUME	요청의 유효하지 않은 볼륨
10015	TRADE_RETCODE_INVALID_PRICE	요청의 유효하지 않은 가격
10016	TRADE_RETCODE_INVALID_STOPS	요청의 유효하지 않은 중지
10017	TRADE_RETCODE_TRADE_DISABLED	거래가 불가능
10018	TRADE_RETCODE_MARKET_CLOSED	마켓이 닫힘
10019	TRADE_RETCODE_NO_MONEY	금액이 부족하여 요청을 완료할 수 없습니다
10020	TRADE_RETCODE_PRICE_CHANGED	가격 변경됨
10021	TRADE_RETCODE_PRICE_OFF	요청을 진행할 시세가 없습니다
10022	TRADE_RETCODE_INVALID_EXPIRATION	요청의 유효하지 않은 주문 만료일
10023	TRADE_RETCODE_ORDER_CHANGED	주문 상태 변경됨
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	너무 빈번한 요청
10025	TRADE_RETCODE_NO_CHANGES	요청에 변경사항 없음
10026	TRADE_RETCODE_SERVER_DISABLES_AT	서버에서 자동거래 사용하지 않음
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	클라이언트 터미널에서 자동거래 사용하지 않음
10028	TRADE_RETCODE_LOCKED	요청을 처리하기 위해 잠김
10029	TRADE_RETCODE_FROZEN	주문 또는 포지션이 동결됨
10030	TRADE_RETCODE_INVALID_FILL	유효하지 않은 주문 채우기 유형

© 2000-2025, MetaQuotes Ltd.

코드	정수	설명
10031	TRADE_RETCODE_CONNECTION	거래 서버와의 연결이 없음
10032	TRADE_RETCODE_ONLY_REAL	활성 계정에 대해서만 작업이 허용됩니다
10033	TRADE_RETCODE_LIMIT_ORDERS	보류 중인 주문의 수가 한도에 도달했습니다
10034	TRADE_RETCODE_LIMIT_VOLUME	기호에 대한 주문 및 포지션의 양이 한계에 도달했습니다
10035	TRADE_RETCODE_INVALID_ORDER	잘못되었거나 금지된 주문 유형
10036	TRADE_RETCODE_POSITION_CLOSED	지정한 POSITION_IDENTIFIER가 있는 포지션이 이미 닫혔습니다
10038	TRADE_RETCODE_INVALID_CLOSE_VOLUME	닫기 볼륨이 현재 포지션 볼륨을 초과합니다
10039	TRADE_RETCODE_CLOSE_ORDER_EXIST	지정 포지션에 대한 마감 순서가 이미 있습니다. 이러한 상황은 위험회피 시스템에서 작업할 때 발생할 수 있습니다: • 포지션에 대한 닫기 명령이 이미 있는 동안 반대 포지션으로 포지션을 닫으려고 할 때 • 이미 존재하는 마감 주문의 총 볼륨과 새로 배치된 주문의 총 볼륨이 현재 위치 볼륨을 초과하는 경우 전체 또는 부분적으로 위치를 닫으려고 할 때
10040	TRADE_RETCODE_LIMIT_POSITIONS	계정에 동시에 존재하는 열려 있는 포지션 수는 서버 설정에 의해 제한될 수 있습니다. 제한에 도달하면 서버는 주문을 시도할 때 TRADE_RETCODE_LIMIT_POSITIONS 오류를 반환합니다. 포지션 계정 유형에 따라 제한이 다르게 작동합니다: • Netting — 열린 포지션의 수를 고려합니다. 제한에 도달하면 플랫폼이 새 주문을 작성할 수 없도록 하고, 이 주문이 실행되면 열린 포지션 수가 증가할 수 있습니다. 실제로 플랫폼에서는 이미 열린 포지션이 있는 기호에 대해서만 주문을 할 수 있습니다. 현재 보류 중인 주문은 실행이 현재 위치의 변경으로 이어질 수 있지만 수를 늘릴 수는 없기 때문에 고려되지 않습니다. • Hedging — 보류 중인 주문 활성화는 항상 새 포지션을 열기 때문에 보류 중인 주문은 열린 포지션과 함께 고려됩니다. 한도에 도달하면 플랫폼에서는 신규 시장 주문과 보류 중인 주문을 모두 배치할 수 없습니다.
10041	TRADE_RETCODE_REJECT_CANCEL	보류 중인 주문 활성화 요청이 거부되어 주문이 취소되었습니다

© 2000-2025, MetaQuotes Ltd.

코드	정수	설명
10042	TRADE_RETCODE_LONG_ONLY	기호에 대해 "장기 포지션만 허용" 규칙이 설정되었기 때문에 요청이 거부되었습니다 (POSITION_TYPE_BUY)
10043	TRADE_RETCODE_SHORT_ONLY	기호에 대해 "단기 포지션만 허용" 규칙이 설정되었기 때문에 요청이 거부되었습니다 (POSITION_TYPE_SELL)
10044	TRADE_RETCODE_CLOSE_ONLY	기호에 대해 "포지션 닫기만 허용" 규칙이 설정되었기 때문에 요청이 거부되었습니다
10045	TRADE_RETCODE_FIFO_CLOSE	"포지션 마감은 FIFO 규칙에 의해서만 허용됨" 플래그가 설정이 거래 계정 (ACCOUNT_FIFO_CLOSE=true)에 대해 되었기 때문에 요청이 거부되었습니다
10046	TRADE_RETCODE_HEDGE_PROHIBITED	거래 계정에 대해 "단일 기호의 반대 포지션이 비활성화됨" 규칙이 설정되었기 때문에 요청이 거부됩니다. 예를 들어, 계정에 매수 포지션이 있는 경우 사용자는 매도 포지션을 열거나 보류 중인 매도 주문을 작성할 수 없습니다. 이 규칙은 위험회피 회계 시스템이 있는 계정에만 적용됩니다 (ACCOUNT_MARGIN_MODE=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING).

© 2000-2025, MetaQuotes Ltd.

컴파일러 경고

컴파일러 경고는 정보 제공 목적으로만 표시되며 오류 메시지는 아닙니다

코드	설명
21	날짜 시간 문자열의 날짜 레코드가 불완전합니다
22	날짜 시간 문자열에 잘못된 날짜 숫자. 요구 사항: 년도 1970 <= X <= 3000 월 0 <X <= 12 일 0 <X <= 31/30/28 (29)….
23	날짜 시간 문자열의 잘못된 시간 숫자 요구 사항: 시 0 <= X <24 분 0 <= X <60
24	RGB 형식의 잘못된 색상: RGB 성분 중 하나가 0보다 작거나 255보다 큼
25	이스케이프 시권스의 알 수 없는 문자 알려진 문자: ₩n ₩r ₩t ₩' ₩" ₩\ ₩x
26	함수의 로컬 변수가 너무 많습니다 (> 512Kb). 수를 줄이십시오
29	열거가 이미 정의됨(중복) - 멤버가 첫 번째 정의에 추가됩니다
30	매크로 재정의
31	변수가 선언되었지만 아무 곳에서도 사용되지 않습니다
32	생성자는 보이드 유형이어야 합니다
33	소멸자는 보이드 유형이어야 합니다
34	상수가 정수 범위 (X> _UI64_MAX | | X <_I64_MIN)에 맞지 않으며 double 유형으로 변환됩니다
35	너무 긴 HEX - 16자 이상의 중요 문자 (자세한 니블이 잘림)
36	HEX 문자열 "0x"에 니블이 없음
37	함수 없음 - 수행할 항목이 없음
38	초기화되지 않은 변수가 사용됩니다
41	함수에 본문이 없으며 호출되지 않습니다
43	형식 캐스팅에서 데이터가 손실될 수 있습니다. 예: int x = (double) z;
44	상수를 변환할 때 (데이터) 정확도가 손실됩니다 예: int x = M_PI
45	비교 연산의 피연산자 부호 간 차이. 예: (char) c1> (uchar) c2
46	함수 가져오기 중 문제 - #import 선언이 필요하거나 함수 가져오기가 종료되었습니다
47	설명이 너무 큼 - 추가 문자는 실행 파일에 포함되지 않습니다
48	선언된 지표 버퍼 수가 필요한 개수보다 적습니다
49	지표에 그래픽 시리즈를 표시할 색상 없음

© 2000-2025, MetaQuotes Ltd.

코드	설명
50	지표를 그릴 그래픽 영상 시리즈 없음
51	스크립트에서 'OnStart' 처리기 함수를 찾을 수 없습니다
52	'OnStart' 처리기 함수가 잘못된 매개변수로 정의되었습니다
53	'OnStart' 함수는 스크립트에서만 정의할 수 있습니다
54	'OnInit' 함수가 잘못된 매개변수로 정의되었습니다
55	'OnInit' 함수는 스크립트에서 사용되지 않습니다
56	'OnDeinit' 함수가 잘못된 매개변수로 정의되었습니다
57	'OnDeinit' 함수는 스크립트에서 사용되지 않습니다
58	두 'OnCalculate' 함수가 정의됩니다. OnCalculate () 단일 간격 배열이 사용됩니다
59	복합 정수 상수를 계산할 때 과다 충전이 감지됨
60	변수가 초기화되지 않았을 수도 있습니다.
61	이 선언으로 인해 지정된 라인에 선언된 로컬 변수를 참조할 수 없습니다
62	이 선언으로 인해 지정된 라인에 선언된 글로벌 변수를 참조할 수 없습니다
63	정적 할당된 배열에 사용할 수 없습니다
64	이 변수 선언은 사전정의된 변수를 숨깁니다
65	식의 값은 항상 true/false입니다
66	수학적 연산에서 변수 또는 bool 유형 식을 사용하는 것은 안전하지 않습니다
67	부호 없는 ulong 유형에서 단원 빼기 연산자를 적용한 결과가 정의되지 않았습니다
68	#property 버전 속성에 지정된 버전이 마켓에서 허용되지 않습니다섹션; 올바른 형식의 #property 버전 ID "XXX.YYY" 입니다
69	빈 제어 문을 찾았습니다
70	이벤트 처리기 함수를 선언하는 동안 잘못된 함수 반환 유형 또는 잘못된 매개변수
71	암묵적 구조 캐스트가 한 유형에 대해 필요합니다
72	이 선언으로 인해 지정된 문자열에 선언된 클래스의 멤버에 직접 액세스할 수 없습니다. 스코프 분해능 작업을 통해서만 액세스가 가능합니다 ::
73	바이너리 상수가 너무 커서 고차 자리가 잘립니다
74	상속된 클래스의 메서드에 있는 매개변수에 다른 const 수정자가 있습니다. 파생 함수가 다음에 오버로드되었습니다: 상위 함수
75	시프트 비트 방향 작업에서 음수 또는 너무 큰 시프트 값, 실행 결과가 정의되지 않았습니다
76	함수는 값을 반환해야 합니다
77	보이드 함수는 값을 반환합니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
78	일부 제어 경로가 값을 반환하지 않습니다
79	글로벌 범위에서는 식을 사용할 수 없습니다
80	연산자 우선 순위를 확인하여 가능한 오류를 확인합니다. 괄호를 사용하여 우선 순위를 명확히 합니다
81	두 OnCalCulate()가 정의됩니다. OHLC 버전이 사용됩니다
82	Struct에 멤버가 없습니다. 크기가 1바이트에 할당됩니다
83	함수의 반환 값을 확인해야 합니다
84	디버깅을 위해 리소스 지표가 컴파일되었습니다. 그러면 성능이 저하됩니다 성능을 높이려면 지표를 다시 컴파일하십시오
85	문자열의 문자 코드가 너무 큽니다. 0 에서 65535 사이여야 합니다
86	문자열에서 인식할 수 없는 문자
87	지표 창 속성(메인 창 또는 하위 창에 화면표시 설정)이 정의되어 있지 않습니다 속성 #property indicator_chart_window이 적용됩니다

© 2000-2025, MetaQuotes Ltd.

컴파일 오류

MetaEdtior 5는 컴파일 중 내장 컴파일러에서 탐지된 프로그램 오류에 대한 오류 매시지를 표시합니다. 이러한 오류 목록은 아래 표에 나와 있습니다. 소스 코드를 실행 파일로 컴파일 하려면 **F7**을 누르십시오. 오류가 포함된 프로그램은 컴파일러에 의해 식별된 오류를 제거할 때까지 컴파일 할 수 없습니다.

코드	설명
100	파일 읽기 오류
101	* 열기 오류. EX5 쓰기
103	사용가능한 메모리가 부족하여 컴파일을 완료할 수 없습니다
104	컴파일러에서 빈 구문 단위를 인식할 수 없습니다
105	#include에 잘못된 파일 이름
106	#include의 파일 엑세스에 오류 (파일이 존재하지 않을 수 있음)
108	#define에 대한 부적절한 이름
109	전처리기의 알 수 없는 명령 (유효한 #include, #define, #property, #import)
110	컴파일러에 알 수 없는 기호
111	기능이 구현되지 않음 (설명이 있지만 본문이 없음)
112	큰 따옴표(") 생략
113	여는 꺽쇠괄호(<) 또는 큰 따옴표(") 생략
114	작은 따옴표(') 생략
115	닫는 꺽쇠괄호 ">" 생략
116	선언에 유형이 지정되지 않았습니다
117	구현의 일부 분기에 반환 연산자 또는 return이 없습니다
118	호출 매개변수의 여는 대괄호가 필요
119	EX5 쓰기 오류
120	배열에 대한 유효하지 않은 액세스
121	함수가 보이드 유형이 아니므로 반환 연산자가 값을 반환해야 합니다
122	소멸자의 잘못된 선언
123	콜론 ":"이 없습니다
124	변수가 이미 선언되었습니다
125	이러한 식별자가 이미 선언된 변수
126	변수 이름이 너무 깁니다 (> 250자)
127	이러한 식별자가 이미 정의된 구조
128	구조가 정의되지 않았습니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
129	이름이 같은 구조 멤버가 이미 정의 되었습니다
130	해당 구조 멤버가 없습니다
131	괄호 쌍이 깨졌습니다
132	여는 괄호 "(" 필요
133	중괄호 불균형 ("}" 없음)
134	컴파일하기 어려움(분기가 너무 많아 내부 스택 레벨이 과다하게 채워짐)
135	읽기 위해 파일을 여는 중 오류 발생
136	메모리가 부족하여 원본 파일을 메모리에 다운로드할 수 없습니다
137	변수 필요
138	참조를 초기화할 수 없습니다
140	할당 필요(신고 시 표시됨)
141	여는 중괄호 "{" 필요
142	매개변수는 동적 배열 전용일 수 있습니다
143	"보이드" 유형은 사용할 수 없습니다
144	")" 또는 "]"에 대한 쌍이 없습니다. 즉 "(" 또는 "[" 가 없음
145	"(" 또는 "["에 대한 쌍이 없습니다. 즉, ")" 또는 "]"가 없음
146	잘못된 배열 크기
147	매개면수가 너무 많습니다 (> 64)
149	이 토큰은 여기서 사용할 수 없습니다
150	유효하지 않은 작업 사용 (유효하지 않은 피연산자)
151	보이드 유형의 식은 허용되지 않습니다
152	연산자 필요
153	브레이크 오용
154	세미콜론 ";" 필요
155	쉼표 "," 필요
156	구조가 아닌 클래스 유형이어야 합니다
157	식 필요
158	HEX 또는 너무 긴 숫자(자리 수>511)에 "HEX 아닌 문자"가 있습니다
159	문자열 상수가 65534 자를 초과합니다
160	여기서는 함수 정의를 사용할 수 없습니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
161	예기치 않은 프로그램 종료
162	구조에 대해 전달 선언이 금지됩니다
163	이 이름의 함수가 이미 정의되어 있고 다른 반환 유형이 있습니다
164	이 이름의 함수가 이미 정의되어 있으며 매개변수 집합이 다릅니다
165	이 이름의 함수가 이미 정의되어 구현되었습니다
166	이 호출에 대한 함수 오버로드를 찾을 수 없습니다
167	반환 값이 보이드 유형인 함수는 값을 반환할 수 없습니다
168	함수가 정의되지 않았습니다
170	값이 필요
171	다음의 *경우:* 식이 정수 상수만 유효한 경우
172	*케이스*(해당 *스위치*)의 값이 이미 사용되고 있습니다
173	정수 필요
174	#mport 에서 식 파일 이름이 필요합니다
175	글로벌 레벨에서는 식을 사용할 수 없습니다
176	";" 앞에 괄호 ")" 가 생략
177	등호 왼쪽에 변수가 있어야 합니다
178	식 결과가 사용되지 않습니다
179	이 *경우* 변수 선언이 허용되지 않습니다
180	문자열에서 숫자로의 암시적 변환
181	숫자를 문자열로 암시적으로 변환
182	오버로드된 함수의 모호한 호출(여러 오버로드 적합)
183	불법 : 적절하지 않은 *경우*
184	유효하지 않은 *경우* 또는 *기본값* (*스위치* 없음)
185	줄임표의 부적절한 사용
186	초기화 단계에 초기화 변수보다 많은 요소가 있습니다
187	상수 (예상 *사례*)
188	상수 표현식이 필요합니다
189	상수 변수는 변경할 수 없습니다
190	닫는 대괄호 또는 쉼표가 필요합니다(배열 멤버 선언)
191	열거자 식별자가 이미 정의되었습니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
192	열거형에는 액세스 한정자(const,extern,static)를 사용할 수 없습니다
193	열거형 멤버가 이미 다른 값으로 선언되었습니다
194	동일한 이름으로 정의된 변수가 있습니다
195	동일한 이름으로 정의된 구조가 있습니다
196	열거형 멤버의 이름이 필요합니다
197	정수 식 필요
198	상수 식에서 0으로 나누기
199	잘못된 함수의 매개변수 수
200	참조별 매개변수는 변수여야 합니다
201	참조를 통과할 동일한 유형의 변수가 필요
202	상수 변수를 비상수 참조로 전달할 수 없습니다
203	양의 정수 상수가 필요
204	보호된 클래스 멤버에 액세스 실패
205	가져오기가 다른 방법으로 이미 정의되었습니다
208	실행 파일이 생성되지 않았습니다
209	지표에 대한 'OnCalculate' 진입점을 찾을 수 없습니다
210	계속 작업은 루프 내부에서만 사용할 수 있습니다
211	비공개(닫힌) 클래스 멤버에 액세스 중 오류
213	구조 또는 클래스 메서드이 선언되지 않았습니다
214	비공개(닫힌) 클래스 메서드에 액세스 중 오류
216	개체를 사용하여 구조를 복사할 수 없습니다
218	인덱스가 매열 범위를 벗어남
219	구조 또는 클래스 선언에서 배열 초기화가 허용되지 않습니다
220	클래스 생성자는 매개변수를 가질 수 없습니다
221	클래스 소멸자에 매개변수를 사용할 수 없습니다
222	이름과 매개변수가 같은 클래스 메서드 또는 구조가 이미 선언되었습니다
223	피연산자 필요
224	이름이 같지만 매개변수가 다른 클래스 메서드 또는 구조가 있습니다 (declaration!=implementation)
225	가져온 함수가 설명되지 않았습니다
226	ZeroMemory()는 보호된 멤버 또는 상속이 있는 개체에는 허용되지 않습니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
227	오버로드된 함수의 모호한 호출 (여러 오버로드에 대한 매개변수의 정확한 일치)
228	변수 이름 필요
229	이 플레이스에서 참조를 선언할 수 없습니다
230	열거 이름으로 이미 사용됨
232	클래스 또는 구조 필요
235	배열을 삭제하기 위해 'delete' 연산자를 호출할 수 없습니다
236	연산자 'while' 필요
237	'delete' 연산자에 포인터가 있어야 합니다
238	이 'switch'에 대한 'default'가 이미 있습니다
239	구문 오류
240	이스케이프 시퀀스는 문자열(₩로 시작)에서만 발생할 수 있습니다
241	배열 필요 -대괄호 '['가 배열에 적용되지 않거나 배열이 아닌 배열이 배열 매개변수로 전달됩니다
242	초기화 시퀀스를 통해 초기화할 수 없습니다
243	가져오기가 정의되지 않았습니다
244	구문 트리의 옵티마이저 오류
245	너무 많은 구조 선언(프로그램 단순화 시도)
246	매개변수를 변환할 수 없습니다
247	'delete' 연산자를 잘못 사용함
248	참조에 대한 포인터를 선언할 수 있습니다
249	참조에 대한 참조를 선언할 수 없습니다
250	포인터에 대한 포인터를 선언할 수 없습니다
251	매개변수 목록에서 구조 선언이 허용되지 않습니다
252	유효하지 않은 형식캐스팅 작업
253	포인터는 클래스 또는 구조물에 대해서만 선언할 수 있습니다
256	신고되지 않은 식별자
257	실행 파일 코드 최적화 도구 오류
258	실행 파일 코드 생성 오류
260	유효하지 않은 'switch' 연산자에 대한 식
261	문자열 상수 풀이 채워짐, 프로그램 단순화
262	열거형으로 변환할 수 없습니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
263	데이터에 'virtual' 사용 안함(클래스 또는 구조의 멤버)
264	보호된 클래스 메서드를 호출할 수 없습니다
265	재정의된 가상 함수가 다른 유형을 반환
266	클래스는 구조에서 상속할 수 없습니다
267	구조를 클래스에서 상속할 수 없습니다
268	생성자는 가상일 수 없습니다 (*가상* 지정자는 허용되지 않음)
269	구조 메서드는 가상일 수 없습니다
270	함수에 본문이 있어야 합니다
271	시스템 기능(터미널 기능)의 오버로딩이 금지됩니다
272	클래스 또는 구조의 멤버가 아닌 함수에 대해 *Const* 지정자가 유효하지 않습니다
274	상수 메서드에서 클래스 멤버를 변경할 수 없습니다
276	부적절한 초기화 순서
277	매개변수의 기본값 누락(기본 매개변수의 특정 선언)
278	기본 매개변수 재정의(선언 및 구현에서 다른 값)
279	상수 개체에 대해 상수 메서드를 호출할 수 없습니다
280	멤버에 엑세스하기 위해 개체가 필요합니다 (비클래스/구조에 대한 점이 지정됨)
281	이미 선언된 구조의 이름을 선언에 사용할 수 없습니다
284	인증되지 않은 변환(마감 상속 시)
285	구조 및 배열을 입력 변수로 사용할 수 없습니다
286	*Const* 지정자가 생성자/소멸자에 대해 잘못되었습니다
287	날짜 시간에 대한 문자열 식이 잘못되었습니다
288	알 수 없는 속성 (#property)
289	잘못된 속성 값
290	#property의 잘못된 속성 인덱스
291	호출 매개변수 생략 -<func (x,)>
293	개체는 참조를 통해 전달되어야 합니다
294	배열은 참조를 통해 전달되어야 합니다
295	함수를 내보낼 수 있는 것으로 선언했습니다
296	함수를 내보낼 수 있는 것으로 선언하지 않았습니다
297	가져온 기능을 내보낼 수 없습니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
298	가져온 함수에 이 매개변수를 사요할 수 없습니다 (포인터, 클래스 또는 동적 배열, 포인터, 클래스 등을 포함하는 구조 전달 금지)
299	클래스여야 합니다
300	#mport가 마감되지 않았습니다
302	유형 불일치
303	외부 변수가 이미 초기화되었습니다
304	내보낸 함수 또는 진입점을 찾을 수 없습니다
305	명시적 생성자 호출이 허용되지 않습니다
306	메서드가 다음으로 선언되었습니다: 상수
307	메서드가 다음으로 선언되지 않았습니다: 상수
308	잘못된 리소스 파일의 크기
309	잘못된 리소스 이름
310	리소스 파일 열기 오류
311	리소스 파일 읽기 오류
312	알 수 없는 리소스 유형
313	잘못된 리소스 파일의 경로
314	지정한 리소스 이름이 이미 사용되고 있습니다
315	함수 유사 매크로에 대해 인수가 필요
316	매크로 정의에서 예상치 못한 기호
317	매크로의 형식 매개변수 오류
318	유효하지 않은 매크로에 대한 매개변수 수
319	매크로에 대한 매개변수가 너무 많습니다
320	너무 복잡합니다. 매크로를 단순화하십시오
321	EnumToString()에 대한 매개변수는 열거형일 수 있습니다
322	리소스 이름이 너무 깁니다
323	지원되지 않는 이미지 형식(24비트 또는 42비트 색상 깊이의 BMP만 지원됨)
324	연산자에 배열을 선언할 수 없습니다
325	함수는 글로벌 범위에서만 선언할 수 있습니다
326	현재 범위에 선언이 허용되지 않습니다
327	로컬 변수 값을 사용하여 정적 변수를 초기화할 수 없습니다
328	기본 생성자가 없는 개체 배열의 잘못된 선언

© 2000-2025, MetaQuotes Ltd.

코드	설명
329	초기화 리스트는 생성자에게만 허용됩니다
330	초기화 목록 후 함수 정의 없음
331	초기화 목록이 비어있습니다
332	생성자에서 배열 초기화가 허용되지 않습니다
333	상위 클래스 멤버를 초기화 목록에서 초기화할 수 없습니다
334	정수 유형의 식 필요
335	배열에 필요한 메모리가 최대값을 초과합니다
336	구조에 필요한 메모리가 최대값을 초과합니다
337	글로벌 레벨에서 선언된 변수에 필요한 메모리가 최대값을 초과합니다
338	로컬 변수에 필요한 메모리가 최대값을 초과힙니다
339	생성자가 정의되지 않았습니다
340	유효하지 않은아이콘 파일 이름
341	아이콘 파일을 열 수 없습니다지정된 경로에서
342	아이콘 파일 올바르지 않으며 ICO가 아닙니다형식
343	클래스/구조 생성자에서 초기화 목록을 사용하여 멤버 재초기화
344	정적 멤버를 생성자 초기화 목록에서 초기화할 수 없습니다
345	클래스/구조의 정적 아닌 멤버를 글로벌 레벨에서 초기화할 수 없습니다
346	클래스/구조 메서드의 이름이 이전에 선언된 멤버의 이름과 일치합니다
347	클래스/구조 멤버의 이름이 이전에 선언한 메서드의 이름과 일치합니다
348	가상 함수를 정적 함수로 선언할 수 없습니다
349	const 한정자는 정적 함수에 허용되지 않습니다
350	생성자 또는 소멸자는 정적일 수 없습니다
351	클래스 또는 구조의 정적 아닌 멤버/메서드는 정적 함수에 액세스할 수 없습니다
352	오버로드 작업 (+, -, [], ++, — etc.)이 연산자 키워드 뒤에 필요합니다
353	일부 작업은 MQL5에서 오버로드할 수 없습니다
354	정의가 선언과 일치하지 않습니다
355	유효하지 않은 매개변수 수를 연산자에 지정했습니다
356	이벤트 핸들링 함수를 찾지 못했습니다
357	메서드를 내보내기 할 수 없습니다
358	상수 개체에 대한 포인터는 상수 개체에 의해 정규화될 수 없습니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
359	클래스 템플릿은 아직 지원되지 않습니다
360	함수 템플릿 오버로드는 아직 지원되지 않습니다
361	함수 템플릿을 적용할 수 없습니다
362	함수 템플릿의 모호한 매개변수(여러 매개변수 유형을 적용할 수 있음)
363	함수 템플릿 인수를 정규화해야 하는 매개변수 유형을 확인할 수 없습니다
364	함수 템플릿의 매개변수 수가 잘못되었습니다
365	함수 템플릿은 가상일 수 없습니다
366	함수 템플릿을 내보낼 수 없습니다
367	함수 템플릿을 가져올 수 없습니다
368	개체를 포함하는 구조는 허용되지 않습니다
369	개체를 포함하는 문자열 배열 및 구조는 허용되지 않습니다
370	정적 클래스/구조 멤버를 명시적으로 초기화해야 합니다
371	컴파일러 제한: 문자열은 65 535자를 초과할 수 없습니다
372	불일치 #ifdef/#endif
373	클래스 개체를 반환할 수 없습니다. 복사 생성자를 찾을 수 없습니다
374	정적 아닌 멤버 및 메서드를 사용할 수 없습니다
375	OnTesterDeinit() 없이 사용할 수 없는 OnTesterInit()
376	형식 매개변수 '%s' 재정의
377	매크로 __FUNCSIG__ 및 __FUNCTION__는 함수 본문 외부에 표시할 수 없습니다
378	유효하지 않은 반환 유형. 예를 들어 구조 또는 포인터를 반환하는 DLL에서 가져온 함수에 대해 이 오류가 발생합니다
379	템플릿 사용 오류
380	사용되지 않음
381	순수 가상 함수를 선언할 때 구문이 잘못되었으며 "=NULL" 또는 "=0"만 허용됩니다
382	순수 지정자("=NULL" 또는 "=0")를 사용하여 가상 함수만 선언할 수 있습니다
383	추상 클래스를 인스턴스화 할 수 없습니다
384	dynamic_cast 연산자를 사용하여 동적 캐스팅의 대상 유형으로 사용자 지정 유형에 대한 포인터를 적용해야 합니다
385	"Pointer to function" 유형이 필요합니다
386	메서드에 대한 포인터는 지원되지 않습니다
387	오류 – 함수로 포인터 유형을 정의할 수 없습니다

© 2000-2025, MetaQuotes Ltd.

코드	설명
388	형식 캐스트를 개인 상속으로 인해 사용할 수 없습니다
389	선언하는 동안 const 한정자가 있는 변수를 초기화해야 합니다
393	공용 액세스권한이 있는 메서드만 인터페이스에 선언될 수 있습니다
394	다른 인터페이스 내부에 유효하지 않은 인터페이스가 중첩됨
395	인터페이스는 다른 인터페이스에서만 파생될 수 있습니다
396	인터페이스 필요
397	인터페이스는 공용 상속만 지원합니다
398	인터페이스는 멤버를 포함할 수 없습니다
399	인터페이스 개체는 직접 만들 수 없으며 상속만 사용합니다
400	지정자를 전달 선언에 사용할 수 없습니다
401	최종 지정자를 사용하여 선언되었으므로 클래스에서 상속할 수 없습니다
402	다음을 재정의할 수 없습니다: 최종 지정자로 선언된 메서드
403	최종 지정자는 가상 기능에만 적용할 수 있습니다
404	재정의 지정자로 표시된 메서드는 실제로 기본 클래스 함수를 재정의 하지 않습니다
405	지정자는 함수를 정의하는 데 사용할 수 없으며 선언에만 사용할 수 있습니다
406	지정된 형식으로 캐스트할 수 없습니다
407	유형은 리소스 변수에 사용할 수 없습니다
408	프로젝트 파일에 오류
409	유니언 멤버로 사용할 수 없습니다
410	이름에 대한 모호한 선택, 사용 컨텍스트를 명시적으로 정의해야 합니다
411	DLL에서 구조를 사용할 수 없습니다
412	삭제 지정자로로 표시된 함수를 호출할 수 없습니다
413	MQL4은 지원되지 않습니다. 이 프로그램을 컴파일하려면 MetaTrader 4 설치 폴더에서 MetaEditor를 사용하십시오

© 2000-2025, MetaQuotes Ltd.

런타임 오류

GetLastError()는 미리 정의된 변수 _LastError에 저장된 마지막 오류 코드를 반환하는 함수입니다. 이 값은 ResetLastError() 함수에 의해 0으로 재설정 할 수 있습니다.

정수	코드	설명
ERR_SUCCESS	0	작업이 성공적으로 완료되었습니다
ERR_INTERNAL_ERROR	4001	예상하지 못한 내부 오류
ERR_WRONG_INTERNAL_PARAMETER	4002	클라이언트 터미널 함수의 내부 호출에 잘못된 매개변수
ERR_INVALID_PARAMETER	4003	시스템 기능을 호출할 때 잘못된 매개변수
ERR_NOT_ENOUGH_MEMORY	4004	메모리가 부족하여 시스템 기능을 수행할 수 없습니다
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	구조에는 문자열 및/또는 동적 배열 개체 및/또는 그러한 개체 및/또는 클래스의 구조가 포함됩니다
ERR_INVALID_ARRAY	4006	잘못된 유형, 잘못된 크기 또는 동적 배열의 손상된 개체 배열
ERR_ARRAY_RESIZE_ERROR	4007	메모리가 부족하여 배열을 재배치하거나 정적 어레이의 크기를 변경하려고 시도하지 못했습니다
ERR_STRING_RESIZE_ERROR	4008	메모리가 부족하여 문자열을 재배치할 수 없습니다
ERR_NOTINITIALIZED_STRING	4009	초기화되지 않은 문자열
ERR_INVALID_DATETIME	4010	유효하지 않은 날짜 및/또는 시간
ERR_ARRAY_BAD_SIZE	4011	배열의 총 요소 양은 2147483647을 초과할 수 없습니다
ERR_INVALID_POINTER	4012	잘못된 포인터
ERR_INVALID_POINTER_TYPE	4013	포인터의 잘못된 유형
ERR_FUNCTION_NOT_ALLOWED	4014	호출에 허용되지 않는 함수
ERR_RESOURCE_NAME_DUPLICATED	4015	동적 리소스와 정적 리소스의 이름이 일치합니다
ERR_RESOURCE_NOT_FOUND	4016	이 이름의 리소스를 EX5에서 찾을 수 없습니다
ERR_RESOURCE_UNSUPPORTED_TYPE	4017	지원되지 않는 리소스 유형 또는 해당 크기가 16 Mb를 초과합니다
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	리소스 이름이 63자를 초과합니다

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_MATH_OVERFLOW	4019	산술 함수를 계산할 때 오버플로우가 발생했습니다
ERR_SLEEP_ERROR	4020	Sleep() 호출 후 테스트 종료 날짜 종료
ERR_PROGRAM_STOPPED	4022	외부에서 테스트가 강제로 중지되었습니다. 예를 들어 최적화가 중단되거나, 시각적 테스트 창이 닫히거나, 테스트 에이전트가 중지되었습니다
ERR_INVALID_TYPE	4023	잘못된 유형
ERR_INVALID_HANDLE	4024	잘못된 핸들
ERR_TOO_MANY_OBJECTS	4025	객체 풀이 채워짐
차트		
ERR_CHART_WRONG_ID	4101	잘못된 차트 ID
ERR_CHART_NO_REPLY	4102	차트가 응답하지 않음
ERR_CHART_NOT_FOUND	4103	차트를 찾을 수 없음
ERR_CHART_NO_EXPERT	4104	차트에 이벤트를 처리할 수 있는 엑스퍼트 어드바이저가 없음
ERR_CHART_CANNOT_OPEN	4105	차트 열기 오류
ERR_CHART_CANNOT_CHANGE	4106	차트 기호 및 기간을 변경하는데 실패함
ERR_CHART_WRONG_PARAMETER	4107	차트 작업 함수에 대한 매개변수의 오류 값
ERR_CHART_CANNOT_CREATE_TIMER	4108	타이머 생성 실패
ERR_CHART_WRONG_PROPERTY	4109	잘못된 차트 속성 ID
ERR_CHART_SCREENSHOT_FAILED	4110	스크린샷 생성 오류
ERR_CHART_NAVIGATE_FAILED	4111	차트를 탐색하는 중 오류
ERR_CHART_TEMPLATE_FAILED	4112	템플릿 적용 오류
ERR_CHART_WINDOW_NOT_FOUND	4113	지표가 포함된 하위 창을 찾을 수 없습니다
ERR_CHART_INDICATOR_CANNOT_ADD	4114	차트에 지표 추가 오류
ERR_CHART_INDICATOR_CANNOT_DEL	4115	차트 지표 추가 오류
ERR_CHART_INDICATOR_NOT_FOUND	4116	지정된 차트에서 지표를 찾을 수 없습니다
그래픽 개체		

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_OBJECT_ERROR	4201	그래픽 개체 작업 오류
ERR_OBJECT_NOT_FOUND	4202	그래픽 개체를 찾을 수 없습니다
ERR_OBJECT_WRONG_PROPERTY	4203	그래픽 개체 속성의 ID가 잘못되었습니다
ERR_OBJECT_GETDATE_FAILED	4204	값에 해당하는 날짜를 가져올 수 없습니다
ERR_OBJECT_GETVALUE_FAILED	4205	날짜에 해당하는 값을 가져올 수 없습니다
MarketInfo		
ERR_MARKET_UNKNOWN_SYMBOL	4301	알 수 없는 기호
ERR_MARKET_NOT_SELECTED	4302	MarketWatch에서 기호가 선택되지 않았습니다
ERR_MARKET_WRONG_PROPERTY	4303	기호 속성의 잘못된 식별자
ERR_MARKET_LASTTIME_UNKNOWN	4304	마지막 틱 시간을 알 수 없음(틱 없음)
ERR_MARKET_SELECT_ERROR	4305	MarketWatch에서 기호 추가 또는 삭제 오류
ERR_MARKET_SELECT_LIMIT	4306	종합 시세에서 선택한 심볼의 한도를 초과했습니다.
ERR_MARKET_SESSION_INDEX	4307	SymbolInfoSessionQuote/SymbolInfoSessionTrade 함수 호출 시 잘못된 세션 ID
History Access		
ERR_HISTORY_NOT_FOUND	4401	요청 내역을 찾을 수 없음
ERR_HISTORY_WRONG_PROPERTY	4402	내역 속성의 ID가 잘못되었습니다
ERR_HISTORY_TIMEOUT	4403	내역 요청 시간 초과
ERR_HISTORY_BARS_LIMIT	4404	터미널 설정에 의해 제한된 요청된 막대 수
ERR_HISTORY_LOAD_ERRORS	4405	내역 로드 시 여러 오류 발생
ERR_HISTORY_SMALL_BUFFER	4407	수신 배열이 너무 작아 요청된 모든 데이터를 저장할 수 없습니다
Global_Variables		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	클라이언트 터미널의 글로벌 변수를 찾을 수 없습니다

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_GLOBALVARIABLE_EXISTS	4502	이름이 같은 클라이언트 터미널의 글로벌 변수가 이미 있습니다
ERR_GLOBALVARIABLE_NOT_MODIFIED	4503	글로벌 변수가 수정되지 않았습니다
ERR_GLOBALVARIABLE_CANNOTREAD	4504	글로벌 변수 값이 있는 파일을 읽을 수 없습니다
ERR_GLOBALVARIABLE_CANNOTWRITE	4505	글로벌 변수 값을 사용하여 파일을 쓸 수 없습니다
ERR_MAIL_SEND_FAILED	4510	이메일 전송 실패
ERR_PLAY_SOUND_FAILED	4511	사운드 재생 실패
ERR_MQL5_WRONG_PROPERTY	4512	프로그램 속성의 식별자가 잘못되었습니다
ERR_TERMINAL_WRONG_PROPERTY	4513	터미널 속성의 식별자가 잘못되었습니다
ERR_FTP_SEND_FAILED	4514	FTP를 통한 파일 전송 실패
ERR_NOTIFICATION_SEND_FAILED	4515	알림 전송에 실패
ERR_NOTIFICATION_WRONG_PARAMETER	4516	알림 전송 매개변수가 잘못되었습니다. 빈 문자열 또는 NULL이 SendNotification() 함수에 전달되었습니다
ERR_NOTIFICATION_WRONG_SETTINGS	4517	터미널의 잘못된 알림 설정 (ID가 지정되지 않았거나 권한이 설정되지 않음)
ERR_NOTIFICATION_TOO_FREQUENT	4518	너무 잦은 알림 전송
ERR_FTP_NOSERVER	4519	FTP 서버가 지정되지 않았습니다
ERR_FTP_NOLOGIN	4520	FTP 사용자명이 지정되지 않았습니다
ERR_FTP_FILE_ERROR	4521	FTP 서버에서 보낼 MQL5₩Files 디렉토리에서 파일을 찾을 수 없습니다
ERR_FTP_CONNECT_FAILED	4522	FTP 연결 실패
ERR_FTP_CHANGEDIR	4523	서버에서 FTP 경로를 찾을 수 없습니다
사용자 지정 지표 버퍼		
ERR_BUFFERS_NO_MEMORY	4601	메모리가 부족하여 지표 버퍼를 배포할 수 없습니다
ERR_BUFFERS_WRONG_INDEX	4602	잘못된 지표 버퍼 인덱스

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
사용자 지정 지표 속성		
ERR_CUSTOM_WRONG_PROPERTY	4603	잘못된 사용자 지정 지표 속성 ID
계정		
ERR_ACCOUNT_WRONG_PROPERTY	4701	잘못된 계정 속성 ID
ERR_TRADE_WRONG_PROPERTY	4751	잘못된 거래 속성 ID
ERR_TRADE_DISABLED	4752	엑스퍼트 어드바이저에 의한 트레이딩 금지
ERR_TRADE_POSITION_NOT_FOUND	4753	포지션을 찾을 수 없습니다
ERR_TRADE_ORDER_NOT_FOUND	4754	주문을 찾을 수 없습니다
ERR_TRADE_DEAL_NOT_FOUND	4755	딜을 찾을 수 없습니다
ERR_TRADE_SEND_FAILED	4756	거래 요청 전송 실패
ERR_TRADE_CALC_FAILED	4758	이익 또는 마진을 계산하지 못했습니다
지표		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	알 수 없는 기호
ERR_INDICATOR_CANNOT_CREATE	4802	지표가 생성되지 않았습니다
ERR_INDICATOR_NO_MEMORY	4803	지표를 추가할 수 있는 충분한 자금이 없습니다
ERR_INDICATOR_CANNOT_APPLY	4804	지표를 다른 지표에 적용할 수 없습니다
ERR_INDICATOR_CANNOT_ADD	4805	차트에 지표 적용 오류
ERR_INDICATOR_DATA_NOT_FOUND	4806	요청 데이터를 찾을 수 없습니다
ERR_INDICATOR_WRONG_HANDLE	4807	잘못된 지표 핸들
ERR_INDICATOR_WRONG_PARAMETERS	4808	지표 생성 시 잘못된 매개변수의 수
ERR_INDICATOR_PARAMETERS_MISSING	4809	지표 작성 시 매개변수 없음
ERR_INDICATOR_CUSTOM_NAME	4810	배열의 첫 번째 매개변수는 사용자 지정 지표의 이름이어야 합니다
ERR_INDICATOR_PARAMETER_TYPE	4811	지표를 만들 때 배열의 매개변수 유형이 잘못되었습니다
ERR_INDICATOR_WRONG_INDEX	4812	요청한 지표 버퍼의 인덱스가 잘못되었습니다
Depth of Market		

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_BOOKS_CANNOT_ADD	4901	Depth Of Market을 추가할 수 없습니다
ERR_BOOKS_CANNOT_DELETE	4902	Depth Of Market을 제거할 수 없습니다
ERR_BOOKS_CANNOT_GET	4903	Depth Of Market의 데이터를 가져올 수 없습니다
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	Depth Of Market에서 새 데이터를 수신하는 동안 오류가 발생했습니다
파일 작업		
ERR_TOO_MANY_FILES	5001	64개 이상의 파일을 동시에 열 수 없습니다
ERR_WRONG_FILENAME	5002	유효하지 않은 파일 이름
ERR_TOO_LONG_FILENAME	5003	파일 이름이 너무 깁니다
ERR_CANNOT_OPEN_FILE	5004	파일 열기 오류
ERR_FILE_CACHEBUFFER_ERROR	5005	캐시를 읽을 메모리가 부족합니다
ERR_CANNOT_DELETE_FILE	5006	파일 삭제 오류
ERR_INVALID_FILEHANDLE	5007	이 핸들이 있는 파일이 닫혔거나 전혀 열리지 않습니다
ERR_WRONG_FILEHANDLE	5008	잘못된 파일 핸들
ERR_FILE_NOTTOWRITE	5009	쓰기 위해 파일을 열어야 합니다
ERR_FILE_NOTTOREAD	5010	읽기 위해 파일을 열어야 합니다
ERR_FILE_NOTBIN	5011	파일을 바이너리 파일로 열어야 합니다
ERR_FILE_NOTTXT	5012	파일을 텍스트로 열어야 합니다
ERR_FILE_NOTTXTORCSV	5013	파일을 텍스트 또는 CSV로 열어야 합니다
ERR_FILE_NOTCSV	5014	파일을 CSV로 열어야 합니다
ERR_FILE_READERROR	5015	파일 읽기 오류
ERR_FILE_BINSTRINGSIZE	5016	파일이 바이너리 파일로 열렸으므로 문자열 크기를 지정해야 합니다
ERR_INCOMPATIBLE_FILE	5017	텍스트 파일은 문자열 배열이어야 하며, 기타 배열은 바이너리여야 합니다
ERR_FILE_IS_DIRECTORY	5018	파일이 아니라, 디렉토리입니다

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_FILE_NOT_EXIST	5019	파일이 존재하지 않습니다
ERR_FILE_CANNOT_REWRITE	5020	파일을 재쓰기할 수 없습니다
ERR_WRONG_DIRECTORYNAME	5021	잘못된 디렉토리 이름
ERR_DIRECTORY_NOT_EXIST	5022	디렉토리가 존재하지 않습니다
ERR_FILE_ISNOT_DIRECTORY	5023	디렉토리가 아니라, 파일입니다
ERR_CANNOT_DELETE_DIRECTORY	5024	디렉토리를 제거할 수 없습니다
ERR_CANNOT_CLEAN_DIRECTORY	5025	디렉토리 비우기 실패 (하나 이상의 파일이 차단되고 제거 작업이 실패했을 수 있음)
ERR_FILE_WRITEERROR	5026	파일에 리소스를 쓰지 못했습니다
ERR_FILE_ENDOFFILE	5027	파일의 끝에 도달했기 때문에 CSV 파일(FileReadString, FileReadNumber, FileReadDatetime, FileReadBool)에서 다음 데이터를 읽을 수 없습니다
문자열 캐스팅		
ERR_NO_STRING_DATE	5030	문자열에 데이터가 없습니다
ERR_WRONG_STRING_DATE	5031	잘못된 문자열 데이터
ERR_WRONG_STRING_TIME	5032	잘못된 문자열 시간
ERR_STRING_TIME_ERROR	5033	문자열을 날짜로 전환하는데 오류
ERR_STRING_OUT_OF_MEMORY	5034	문자열에 대해 메모리가 충분하지 않습니다
ERR_STRING_SMALL_LEN	5035	문자열 길이가 예상보다 적습니다
ERR_STRING_TOO_BIGNUMBER	5036	너무 큰 수, ULONG_MAX보다 큼
ERR_WRONG_FORMATSTRING	5037	유효하지 않은 서식 문자열
ERR_TOO_MANY_FORMATTERS	5038	서식 지정자의 양이 매개변수보다 큼
ERR_TOO_MANY_PARAMETERS	5039	서식 지정자보다 많은 매개변수 양
ERR_WRONG_STRING_PARAMETER	5040	문자열 유형의 매개변수가 손상됨
ERR_STRINGPOS_OUTOFRANGE	5041	문자열 외부에 위치
ERR_STRING_ZEROADDED	5042	문자열 끝에 0이 추가되어 사용할 수 없는 작업입니다
ERR_STRING_UNKNOWNTYPE	5043	문자열로 변환할 때 알 수 없는 데이터 유형

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_WRONG_STRING_OBJECT	5044	손상된 문자열 개체
배열을 사용한 작업		
ERR_INCOMPATIBLE_ARRAYS	5050	호환되지 않는 배열을 복사. 문자열 배열은 문자열 배열 및 숫자 배열은 숫자 배열에만 복사할 수 있습니다
ERR_SMALL_ASSERIES_ARRAY	5051	수신 배열이 AS_SERIES로 선언되었으며 크기가 부족합니다
ERR_SMALL_ARRAY	5052	시작 포지션이 배열 외부에 있는 너무 작은 배열입니다
ERR_ZEROSIZE_ARRAY	5053	0길이의 배열
ERR_NUMBER_ARRAYS_ONLY	5054	숫자 배열이어야 합니다
ERR_ONEDIM_ARRAYS_ONLY	5055	1차원 배열이어야 합니다
ERR_SERIES_ARRAY	5056	시계열을 사용할 수 없습니다
ERR_DOUBLE_ARRAY_ONLY	5057	double 유형의 배열이어야 합니다
ERR_FLOAT_ARRAY_ONLY	5058	float 유형의 배열이어야 합니다
ERR_LONG_ARRAY_ONLY	5059	long 유형의 배열이어야 합니다
ERR_INT_ARRAY_ONLY	5060	int 유형의 배열이어야 합니다
ERR_SHORT_ARRAY_ONLY	5061	short 유형의 배열이어야 합니다
ERR_CHAR_ARRAY_ONLY	5062	char 유형의 배열이어야 합니다
ERR_STRING_ARRAY_ONLY	5063	문자열 배열 전용
OpenCL을 사용한 작업		
ERR_OPENCL_NOT_SUPPORTED	5100	OpenCL 기능은 이 컴퓨터에서 지원되지 않습니다
ERR_OPENCL_INTERNAL	5101	다음을 실행하는 동안 내부 오류가 발생했습니다: OpenCL 실행 중
ERR_OPENCL_INVALID_HANDLE	5102	유효하지 않은 OpenCL 핸들
ERR_OPENCL_CONTEXT_CREATE	5103	다음을 생성하는데 오류: OpenCL 컨텍스트
ERR_OPENCL_QUEUE_CREATE	5104	OpenCL에서 실행 대기열을 만들지 못했습니다
ERR_OPENCL_PROGRAM_CREATE	5105	다음 중 오류 발생: OpenCL 프로그램 컴파일링
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	너무 긴 커널 이름 (OpenCL 커널)

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_OPENCL_KERNEL_CREATE	5107	다음 생성 시 오류: OpenCL 커널
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	다음 중 오류 발생: OpenCL 커널에 대한매개변수 설정 중
ERR_OPENCL_EXECUTE	5109	OpenCL 프로그램 런타임 오류
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	유효하지 않은 OpenCL 버퍼 크기
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	유효하지 않은 OpenCL 버퍼의 오프셋
ERR_OPENCL_BUFFER_CREATE	5112	다음을 생성하는데 실패: OpenCL 버퍼
ERR_OPENCL_TOO_MANY_OBJECTS	5113	너무 많은 OpenCL 개체
ERR_OPENCL_SELECTDEVICE	5114	OpenCL 장치 선택 오류
데이터베이스 작업		
ERR_DATABASE_INTERNAL	5120	내부 데이터베이스 오류
ERR_DATABASE_INVALID_HANDLE	5121	유효하지 않은 데이터베이스 핸들
ERR_DATABASE_TOO_MANY_OBJECTS	5122	허용되는 최대 데이터베이스 개체 수를 초과
ERR_DATABASE_CONNECT	5123	데이터베이스 연결 오류
ERR_DATABASE_EXECUTE	5124	요청 실행 오류
ERR_DATABASE_PREPARE	5125	요청 생성 오류
ERR_DATABASE_NO_MORE_DATA	5126	읽을 데이터가 더 없습니다
ERR_DATABASE_STEP	5127	다음 요청 진입으로 이동 실패
ERR_DATABASE_NOT_READY	5128	요청 결과를 읽기 위한 데이터가 준비되지 않았습니다
ERR_DATABASE_BIND_PARAMETERS	5129	SQL 요청에 매개변수를 자동 대체하지 못했습니다
WebRequest 작업		
ERR_WEBREQUEST_INVALID_ADDRESS	5200	유효하지 않은 URL
ERR_WEBREQUEST_CONNECT_FAILED	5201	지정 URL로 연결 실패
ERR_WEBREQUEST_TIMEOUT	5202	시간 초과
ERR_WEBREQUEST_REQUEST_FAILED	5203	HTTP 요청 실패
네트워크 작업 (소켓)		
ERR_NETSOCKET_INVALIDHANDLE	5270	유효하지 않은 소켓 핸들이 함수로 전달됨

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_NETSOCKET_TOO_MANY_OPENED	5271	너무 많은 오픈 소캣 (최대 128)
ERR_NETSOCKET_CANNOT_CONNECT	5272	원격 호스트에 연결 실패
ERR_NETSOCKET_IO_ERROR	5273	소켓에서 데이터 송수신 실패
ERR_NETSOCKET_HANDSHAKE_FAILED	5274	보안 연결 설정을 실패 (TLS Handshake)
ERR_NETSOCKET_NO_CERTIFICATE	5275	연결을 보호하는 인증서에 데이터가 없음
사용자 지정 기호		
ERR_NOT_CUSTOM_SYMBOL	5300	사용자 지정 기호를 지정해야 합니다
ERR_CUSTOM_SYMBOL_WRONG_NAME	5301	사용자 지정 기호의 이름이 잘못되었습니다. 기호 이름에는 구두점, 공백 또는 특수 문자(may only contain ".", "_", "&" and "#") 없이 라틴 문자만 사용할 수 있습니다. <, >, :, ", /, \, |, ?, *. 문자를 사용하지 마십시오
ERR_CUSTOM_SYMBOL_NAME_LONG	5302	사용자 지정 기호의 이름이 너무 깁니다. 기호 이름의 길이는 끝 0자를 포함하여 32자를 초과할 수 없습니다
ERR_CUSTOM_SYMBOL_PATH_LONG	5303	사용자 지정 기호의 경로가 너무 깁니다. 경로 길이는 "Custom\, 기호 이름, 그룹 구분 기호 및 끝 0을 포함하여 128자를 초과할 수 없습니다
ERR_CUSTOM_SYMBOL_EXIST	5304	이름이 같은 사용자 지정 기호가 이미 있습니다
ERR_CUSTOM_SYMBOL_ERROR	5305	사용자 지정 기호를 작성, 삭제 또는 변경하는 동안 오류가 발생했습니다
ERR_CUSTOM_SYMBOL_SELECTED	5306	마켓 워치(Market Watch)에서 선택한 사용자 지정 기호를 삭제하려고 합니다
ERR_CUSTOM_SYMBOL_PROPERTY_WRONG	5307	유효하지 않은 사용자 지정 기호 속성
ERR_CUSTOM_SYMBOL_PARAMETER_ERROR	5308	사용자 지정 기호의 속성을 설정하는 동안 잘못된 매개변수
ERR_CUSTOM_SYMBOL_PARAMETER_LONG	5309	사용자 지정 기호의 속성을 설정하는 동안 문자열 매개변수가 너무 깁

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
		니다
ERR_CUSTOM_TICKS_WRONG_ORDER	5310	배열의 틱은 시간 순서대로 배열되지 않습니다
이코노믹 캘린더		
ERR_CALENDAR_MORE_DATA	5400	배열의 크기가 부족하여 모든 값에 대한 설명을 수신할 수 없습니다
ERR_CALENDAR_TIMEOUT	5401	요청 시간 제한 초과
ERR_CALENDAR_NO_DATA	5402	국가를 찾을 수 없습니다]
데이터베이스 작업		
ERR_DATABASE_ERROR	5601	일반 오류
ERR_DATABASE_LOGIC	5602	SQLite 내부 로직 오류
ERR_DATABASE_PERM	5603	액세스 거부됨
ERR_DATABASE_ABORT	5604	콜백 루틴 중단이 요청됨
ERR_DATABASE_BUSY	5605	데이터베이스 파일 잠김
ERR_DATABASE_LOCKED	5606	데이터베이스 테이블 잠김
ERR_DATABASE_NOMEM	5607	작업을 완료하기에 충분하지 않은 메모리
ERR_DATABASE_READONLY	5608	읽기전용 데이터베이스에 쓰기 시도
ERR_DATABASE_INTERRUPT	5609	sqlite3_interrupt()가 작업을 종료함
ERR_DATABASE_IOERR	5610	Disk I/O error
ERR_DATABASE_CORRUPT	5611	데이터베이스 디스크 이미지가 손상됨
ERR_DATABASE_NOTFOUND	5612	sqlite3_file_control()의 알 수 없는 작업 코드
ERR_DATABASE_FULL	5613	데이터베이스가 가득 차서 삽입에 실패
ERR_DATABASE_CANTOPEN	5614	데이터베이스 파일을 열 수 없습니다
ERR_DATABASE_PROTOCOL	5615	데이터베이스 잠금 프로토콜 오류
ERR_DATABASE_EMPTY	5616	내부 사용 전용
ERR_DATABASE_SCHEMA	5617	데이터베이스 개요 변경됨

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_DATABASE_TOOBIG	5618	문자열 또는 BLOB가 크기 제한을 초과
ERR_DATABASE_CONSTRAINT	5619	제약 위반으로 인해 중단
ERR_DATABASE_MISMATCH	5620	데이터 유형 일치하지 않음
ERR_DATABASE_MISUSE	5621	라이브러리가 잘못 사용됨
ERR_DATABASE_NOLFS	5622	호스트에서 지원되지 않는 OS 기능 사용
ERR_DATABASE_AUTH	5623	인증 거부됨
ERR_DATABASE_FORMAT	5624	사용되지 않음
ERR_DATABASE_RANGE	5625	바인딩 매개변수 오류, 잘못된 인덱스
ERR_DATABASE_NOTADB	5626	데이터베이스 파일이 아닌 열려 있는 파일
행렬 및 벡터 메서드		
ERR_MATRIX_INTERNAL	5700	행렬/벡터 실행 하위 시스템의 내부 오류
ERR_MATRIX_NOT_INITIALIZED	5701	행렬/벡터의 다음이 되지 않음 초기화
ERR_MATRIX_INCONSISTENT	5702	작동 중인 행렬/벡터의 크기가 일관되지 않음
ERR_MATRIX_INVALID_SIZE	5703	유효하지 않은 행렬/벡터 크기
ERR_MATRIX_INVALID_TYPE	5704	유효하지 않은 행렬/벡터 유형
ERR_MATRIX_FUNC_NOT_ALLOWED	5705	이 행렬/벡터에 사용할 수 없는 함수
ERR_MATRIX_CONTAINS_NAN	5706	행렬/벡터에 숫자가 아닌 값이 포함됨(Nan/Inf)
ONNX models		
ERR_ONNX_INTERNAL	5800	ONNX 내부 오류
ERR_ONNX_NOT_INITIALIZED	5801	ONNX 런타임 API 초기화 오류
ERR_ONNX_NOT_SUPPORTED	5802	MQL5에서 지원하지 않는 속성 또는 값
ERR_ONNX_RUN_FAILED	5803	ONNX 런타임 API 실행 오류
ERR_ONNX_INVALID_PARAMETERS_COUNT	5804	OnnxRun에 전달된 잘못된 매개변수 수
ERR_ONNX_INVALID_PARAMETER	5805	잘못된 매개변수 값

© 2000-2025, MetaQuotes Ltd.

정수	코드	설명
ERR_ONNX_INVALID_PARAMETER_TYPE	5806	잘못된 매개변수 유형
ERR_ONNX_INVALID_PARAMETER_SIZE	5807	잘못된 매개변수 사이즈
ERR_ONNX_WRONG_DIMENSION	5808	텐서 차원이 설정되지 않았거나 유효하지 않음
User-Defined Errors		
ERR_USER_ERROR_FIRST	65536	사용자 정의 오류는 이 코드로 시작합니다

추가 참조

거래 서버 반환 코드

© 2000-2025, MetaQuotes Ltd.

입력 및 출력 상수

상수:

- 파일 열기 플래그
- 파일 속성
- 파일 내부 위치 지정
- 코드 페이지 사용법
- MessageBox

© 2000-2025, MetaQuotes Ltd.

파일 열기 플래그

파일 열기 플래그 값은 파일 액세스 모드를 지정합니다. 플래그는 다음과 같이 정의됩니다:

식별자	값	설명
FILE_READ	1	파일을 읽기 위해 열려 있습니다. FileOpen()에서 플래그가 사용됩니다. FILE_WRITE 및/또는 FILE_READ의 파일 사양이 필요합니다.
FILE_WRITE	2	파일이 쓰기 위해 열려 있습니다. FileOpen()에서 플래그가 사용됩니다. FILE_WRITE 및/또는 FILE_READ의 파일 사양이 필요합니다.
FILE_BIN	4	바이너리 읽기/쓰기 모드(문자열이 문자열로 변환되지 않음). FileOpen()에서 플래그가 사용됩니다.
FILE_CSV	8	CSV 파일 (모든 요소가 해당 유형, 유니코드 또는 ANSI의 문자열로 변환되고 구분 기호로 구분됨). FileOpen()에서 플래그가 사용됩니다.
FILE_TXT	16	단순 텍스트 파일 (csv 파일과 동일하지만 구분자를 고려하지 않음). FileOpen()에서 플래그가 사용됩니다.
FILE_ANSI	32	Strings of ANSI 유형의 문자열 (1바이트 기호). FileOpen()에서 플래그가 사용됩니다.
FILE_UNICODE	64	UNICODE 유형의 문자열 (2바이트 기호). FileOpen()에서 플래그가 사용됩니다.
FILE_SHARE_READ	128	여러 프로그램에서 읽기 위한 공유 액세스. 플래그는 FileOpen()에서 사용되지만, 파일을 열 때 FILE_WRITE 및/또는 FILE_READ 플래그를 나타낼 필요는 없습니다.
FILE_SHARE_WRITE	256	여러 프로그램에서 쓰기 위한 공유 액세스. 플래그는 FileOpen()에서 사용되지만, 파일을 열 때 FILE_WRITE 및/또는 FILE_READ 플래그를 나타낼 필요는 없습니다.
FILE_REWRITE	512	FileCopy() 및 FileMove() 함수를 사용하여 파일을 다시 쓸 수 있습니다. 파일이 존재하거나 쓰기 위해 열려 있어야 하며, 그렇지 않으면 파일이 열리지 않습니다.
FILE_COMMON	4096	모든 클라이언트 터미널의 공용 폴더에 있는 파일 경로 \Terminal\Common\Files. 플래그는 FileOpen(), FileCopy(), FileMove() 및 FileIsExist() 함수에 사용됩니다.

파일을 열 때 하나 이상의 플래그를 지정할 수 있습니다. 이는 플래그의 조합입니다. 플래그 조합은 열거된 플래그 사이에 위치한 로직 부호 OR (|)를 사용하여 작성됩니다. 예를 들어 읽기 및 쓰기를 위해 CSV 형식의 파일을 동시에 열려면 FILE_READ|FILE_WRITE|FILE_CSV 조합을 지정합니다.

예:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

읽기 및 쓰기 플래그를 지정할 때 몇 가지 특정 작업 기능이 있습니다:

© 2000-2025, MetaQuotes Ltd.

- FILE_READ가 지정된 경우 기존 파일을 열려고 시도합니다. 파일이 없는 경우 파일을 열지 못하고 새 파일이 작성되지 않습니다.
- FILE_READ|FILE_WRITE - 지정된 이름의 파일이 없는 경우 새 파일이 생성됩니다.
- FILE_WRITE - 파일이 0 크기로 다시 생성됩니다.

파일을 열 때 FILE_WRITE 및/또는 FILE_READ의 규격이 필요합니다.

열린 파일의 읽기 유형을 정의하는 플래그는 우선 순위를 가집니다. 가장 높은 플래그는 FILE_CSV이고, 다음으로 FILE_BIN이 되며 FILE_TXT는 우선순위가 가장 낮습니다. 따라서 동시에 여러 플래그가 지정되면, (FILE_TXT|FILE_CSV 또는 FILE_TXT|FILE_BIN 또는 FILE_BIN|FILE_CSV) 우선순위가 가장 높은 플래그가 사용됩니다.

인코딩 유형을 정의하는 플래그에도 우선 순위가 있습니다. FILE_UNICODE가 FILE_ANSI보다 우선 순위가 높습니다. 따라서 FILE_UNICODE|FILE_ANSI 조합을 지정하면 FILE_UNICODE 플래그가 사용됩니다.

FILE_UNICODE 또는 FILE_ANSI가 표시되지 않으면 FILE_UNICODE 가 암시됩니다. FILE_CSV, FILE_BIN, FILE_TXT를 지정하지 않으면 FILE_CSV가 암시됩니다.

텍스트 파일로 읽을 수 있도록 파일이 열려 있고 (FILE_TXT 또는 FILE_CSV), 특수 2바이트 표시 0xff,0xfe 가 시작되는 파일에서 FILE_ANSI가 지정된 경우에도 FILE_UNICODE가 됩니다.

추가 참조

파일 기능

© 2000-2025, MetaQuotes Ltd.

파일 속성

파일 속성을 가져오기 위해 FileGetInteger() 함수를 사용합니다. 호출 중에 ENUM_FILE_PROPERTY_INTEGER 열거에서 필요한 속성 식별자가 전달됩니다.

ENUM_FILE_PROPERTY_INTEGER

ID	ID 설명
FILE_EXISTS	존재 여부 확인
FILE_CREATE_DATE	생성일
FILE_MODIFY_DATE	마지막 수정일
FILE_ACCESS_DATE	파일 마지막 접근일
FILE_SIZE	파일 크기(바이트)
FILE_POSITION	파일에서 포인터의 위치
FILE_END	파일 부호 끝 가져오기
FILE_LINE_END	선 부호 끝 가져오기
FILE_IS_COMMON	파일이 모든 터미널의 공유 폴더에 열립니다 (FILE_COMMON 참조)
FILE_IS_TEXT	파일이 텍스트 파일로 열립니다 (FILE_TXT 참조)
FILE_IS_BINARY	파일이 바이너리 파일로 열립니다 (FILE_BIN 참조)
FILE_IS_CSV	파일이 CSV로 열립니다 (FILE_CSV 참조)
FILE_IS_ANSI	파일이 ANSI로 열립니다 (FILE_ANSI 참조)
FILE_IS_READABLE	열린 파일을 읽을 수 있습니다 (FILE_READ 참조)
FILE_IS_WRITABLE	열려 있는 파일에 쓸 수 있습니다 (FILE_WRITE 참조)

FileGetInteger() 함수에 두 가지 다른 호출 옵션이 있습니다. 첫 번째 옵션에서는 파일의 속성을 가져오기 위해 FileOpen() 기능을 사용하여 파일을 여는 동안 파일 핸들이 지정됩니다. 이 옵션을 사용하면 파일의 모든 속성을 가져올 수 있습니다.

FileGetInteger() 함수의 두 번째 옵션은 파일 이름별로 파일 속성값을 반환합니다. 이 옵션을 사용하면 다음과 같은 일반 속성만 얻을 수 있습니다:

- FILE_EXISTS – 지정된 이름의 파일 존재
- FILE_CREATE_DATE – 지정된 이름의 파일 생성 날짜
- FILE_MODIFY_DATE – 지정된 이름의 파일 수정 날짜
- FILE_ACCESS_DATE - 지정된 이름의 파일에 대한 마지막 액세스 날짜
- FILE_SIZE – 지정된 이름의 파일 크기

위에서 지정한 속성 이외의 속성을 가져오려고 하면 FileGetInteger() 호출의 두 번째 옵션이 오류를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

파일 내부 위치 지정

대부분의 파일 기능은 데이터 읽기/쓰기 작업과 연결됩니다. 동시에 FileSeek()을 사용하여 다음 읽기 또는 쓰기 작업을 수행할 파일 내부 우치에 대한 파일 포인터의 위치를 지정할 수 있습니다. ENUM_FILE_POSITION 열거에는 유효한 포인터 위치가 포함되어 있으며, 이 위치를 기준으로 다음 작업의 이동을 바이트 단위로 지정할 수 있습니다.

ENUM_FILE_POSITION

식별자	설명
SEEK_SET	파일 시작
SEEK_CUR	파일 포인터의 현재 위치
SEEK_END	파일 끝

추가 참조

FileIsEnding, FileIsLineEnding

© 2000-2025, MetaQuotes Ltd.

문자열 변환 작업에서 코드페이지 사용

When converting string variables into arrays of char type and back, the encoding that by default corresponds to the current ANSI of Windows operating system (CP_ACP) is used in MQL5. 다른 인코딩 유형을 지정하려면 CharArrayToString(), StringToCharArray() 및 FileOpen() 함수에 대한 추가 매개변수로 설정할 수 있습니다.

다음 표에는 가장 인기 있는 코드페이지의 기본 제공 상수가 나열되어 있습니다. 언급되지 않은 코드페이지는 페이지에 해당하는 코드로 지정할 수 있습니다.

코드페이지의 기본 상수 제공

정수	값	설명
CP_ACP	0	현재 Windows ANSI 코드 페이지.
CP_OEMCP	1	현재 시스템 OEM 코드 페이지.
CP_MACCP	2	현재 시스템 Macintosh 코드 페이지. **참고:** 이 값은 이전에 만든 프로그램 코드에서 주로 사용되며, 현재 Macintosh 컴퓨터는 인코딩에 유니코드를 사용하기 때문에 아무 소용이 없습니다.
CP_THREAD_ACP	3	현재 수레드의 Windows ANSI 코드 페이지.
CP_SYMBOL	42	기호 코드 페이지
CP_UTF7	65000	UTF-7 코드 페이지.
CP_UTF8	65001	UTF-8 코드 페이지.

추가 참조

클라이언트 터미널 속성

© 2000-2025, MetaQuotes Ltd.

MessageBox 대화상자 창의 상수

이 섹션에는 MessageBox() 기능의 반환 코드가 포함되어 있습니다. 메시지 창에 Cancel 버튼이 있으면 ESC 키나 Cancel 버튼을 누르면 IDCANCEL이 반환됩니다. 메시지 창에 Cancel 버튼이 없는 경우 ESC를 눌러도 아무런 효과가 없습니다.

정수	값	설명
IDOK	1	"확인" 버튼을 눌렀습니다
IDCANCEL	2	"취소" 버튼을 눌렀습니다
IDABORT	3	"중단" 버튼을 눌렀습니다
IDRETRY	4	"재시도" 버튼을 눌렀습니다
IDIGNORE	5	"무시" 버튼을 눌렀습니다
IDYES	6	"예" 버튼을 눌렀습니다
IDNO	7	"아니오" 버튼을 눌렀습니다
IDTRYAGAIN	10	"다시 시도" 버튼을 눌렀습니다
IDCONTINUE	11	"계속" 버튼을 눌렀습니다

MessageBox() 함수의 주 플래그는 대화상자 창의 내용과 동작을 정의합니다. 이 값은 다음 플래그 그룹의 조합일 수 있습니다:

정수	값	설명
MB_OK	0x00000000	메시지 창에는 확인 버튼 하나만 있습니다 기본
MB_OKCANCEL	0x00000001	메시지 창에는 확인과 취소 두 가지 버튼이 있습니다
MB_ABORTRETRYIGNORE	0x00000002	메시지 창에는 중단, 재시도 및 무시 세 가지 버튼이 있습니다
MB_YESNOCANCEL	0x00000003	메시지 창에는 예, 아니오 및 취소 세 가지 버튼이 있습니다
MB_YESNO	0x00000004	메시지 창에는 예, 아니오 두 가지 버튼이 있습니다
MB_RETRYCANCEL	0x00000005	메시지 창에는 재시도, 취소 두 가지 버튼이 있습니다
MB_CANCELTRYCONTINUE	0x00000006	메시지 창에는 취소, 다시 시도, 계속 세 가지 버튼이 있습니다

메시지 창에 아이콘을 표시하려면 추가 플래그를 지정해야 합니다:

© 2000-2025, MetaQuotes Ltd.

정수	값	설명
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	STOP 기호 아이콘
MB_ICONQUESTION	0x00000020	물음표 아이콘
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	느낌표/경고 표시 아이콘
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	동그라미 친 i 기호

기본 버튼은 다음 플래그로 정의됩니다:

정수	값	설명
MB_DEFBUTTON1	0x00000000	첫 번째 버튼 MB_DEFBUTTON1가 기본 값입니다 (MB_DEFBUTTON2, MB_DEFBUTTON3, 또는 MB_DEFBUTTON4가 지정되지 않은 경우)
MB_DEFBUTTON2	0x00000100	두 번째 버튼이 기본값입니다
MB_DEFBUTTON3	0x00000200	세 번째 버튼이 기본값입니다
MB_DEFBUTTON4	0x00000300	네 번째 버튼이 기본값입니다

© 2000-2025, MetaQuotes Ltd.

MQL5 프로그램

mql5-프로그램이 작동하려면 컴파일 버튼 또는 F7 키를 컴파일해야 합니다. 컴파일은 오류 없이 통과해야 합니다 (일부 경고는 발생할 수 있으며, 이러한 경고는 분석되어야 합니다). 이 프로세스에서 EX5 확장자가 같은 이름의 실행 파일이 해당 디렉토리 terminal_dir₩MQL5₩Experts, terminal_dir₩MQL5₩Indicators 또는 terminal_dir₩MQL5₩Scripts 에 생성되어야 합니다. 이 파일은 실행할 수 있습니다

MQL5 프로그램의 작동 기능은 다음 섹션에 설명되어 있습니다.

- 프로그램 실행 - 미리 정의된 event-handlers를 호출하는 순서.
- 거래 전략 테스트 - Strategy Tester(전략 테스터)에서 MQL5 프로그램의 작동 기능
- 클라이언트 터미널 이벤트 - 프로그램에서 처리할 수 있는 이벤트에 대한 설명.
- 가져온 함수 호출 - 가져온 함수에 대한 설명 순서, 허용된 매개 변수, 검색 세부 정보 및 통화 계약.
- 런타임 에러 - 런타임 및 심각한 오류에 대한 정보 가져오기.

Navigator 창에서 드래그앤드롭 방식으로 열려 있는 차트 중 하나에 Expert Advisor, 커스텀 지표 및 스크립트를 첨부합니다.

Expert Advisor 작동을 중지하려면 차트에서 해당 관리자를 제거해야 합니다. 차트 상황에 맞는 메뉴에서 "Expert 목록"을 선택한 후 목록에서 Expert Advisor를 선택하고 "제거" 버튼을 클릭합니다. Expert Advisor의 작동은 "AutoTrading" 버튼의 상태에도 영향을 받습니다.

커스텀 지표를 중지하려면 차트에서 해당 지표를 제거해야 합니다.

커스텀 지표와 Expert Advisor는 차트에서 명시적으로 삭제될 때까지 작동하며, 첨부된 Expert Advisors 및 지표에 대한 정보는 클라이언트 터미널 세션 사이에 저장됩니다.

스크립트는 한 번 실행되며 작업이 완료되거나 현재 차트 상태가 변경되거나 클라이언트 터미널이 종료될 때 자동으로 삭제됩니다. 클라이언트 터미널 스크립트에 대한 정보가 저장되지 않기 때문에 클라이언트 터미널 스크립트의 재시작이 시작되지 않습니다.

한 차트에서 최대 1개의 Expert Advisor, 1개의 스크립트 및 무제한의 지표가 작동할 수 있습니다.

서비스는 작업을 위해 차트에 바인딩할 필요가 없으며 보조 기능을 수행하도록 설계되었습니다. 예를 들어 서비스에서 커스텀 심볼을 생성하고 차트를 열고 네트워크 함수를 사용하여 해당 서비스에 대한 데이터를 무한 루프에서 수신한 후 지속적으로 업데이트할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

실행 중인 프로그램

각 스크립트, 각 서비스 및 각 Expert Advisor는 고유한 개별 스레드에서 실행됩니다. 하나의 심볼에 따라 계산된 모든 지표는 서로 다른 차트에 연결되어 있더라도 동일한 스레드에서 작동합니다. 따라서, 하나의 심볼에 있는 모든 지표는 하나의 쓰레드의 자원을 공유합니다.

틱 처리 및 내역 동기화와 같은 심볼와 관련된 다른 모든 작업도 지표와 동일한 스레드에서 일관되게 수행됩니다. 즉, 지표에서 무한 동작을 수행하면 해당 심볼과 관련된 다른 모든 이벤트가 수행되지 않습니다.

Expert Advisor를 실행할 때는 실제 거래 환경이 있고 필요한 기호와 기간의 내역에 액세스 할 수 있는지 확인하고 터미널과 서버 간에 데이터를 동기화합니다. 이러한 모든 절차에서 터미널은 5초 미만의 시작 지연을 제공하며, 그 후 Expert Advisor가 사용 가능한 데이터로 시작됩니다. 따라서 서버에 연결되지 않은 경우, Expert Advisor 시작 시간이 지연될 수 있습니다.

아래 표에는 MQL5 프로그램의 간략한 요약이 나와 있습니다:

Program	Running	참고
Service	별도의 스레드, 서비스의 스레드 수는 서비스 수와 같습니다	루프 서비스가 다른 프로그램의 실행을 중단할 수 없습니다
Script	별도의 스레드, 스크립트의 스레드 수는 스크립트 수와 같습니다	루프된 스크립트는 다른 프로그램의 실행을 중단할 수 없습니다
Expert Advisor	별도의 스레드인 Expert Advisor의 스레드 수는 Expert Advisor의 수와 동일합니다	루프가 있는 Expert Advisor는 다른 프로그램의 실행을 중단할 수 없습니다
Indicator	심볼에 있는 모든 지표에 대한 하나의 스레드. 스레드 수는 지시자가 있는 심볼 수와 같습니다	한 지표에 무한 루프가 있으면 이 심볼에 있는 다른 모든 지표가 중지됩니다

프로그램이 차트에 연결된 직후 클라이언트 터미널 메모리에 업로드되고 글로벌 변수가 초기화됩니다. 클래스 유형의 일부 글로벌 변수에 생성자가 있는 경우, 글로벌 변수를 초기화하는 동안 이 생성자가 호출됩니다.

그 후 프로그램은 클라이언트 터미널에서 이벤트에 대기할 것입니다. 각 mql5-프로그램에는 적어도 하나의 event-handler가 있어야 하며, 그렇지 않으면 로드된 프로그램이 실행되지 않습니다. 이벤트 핸들러에는 미리 정의된 이름, 매개 변수 및 반환 유형이 있습니다.

유형	함수명	Parameter	어플리케이션	코멘트
int	OnInit	none	Expert Advisor 및 지표	Init 이벤트 핸들러. void return 타입을 사용할 수 있습니다.
void	OnDeinit	const int reason	Expert Advisor 및 지표	Deinit 이벤트 핸들러.
void	OnStart	none	스크립트 및 서비스	Start 이벤트 핸들러.

© 2000-2025, MetaQuotes Ltd.

유형	함수명	Parameter	어플리케이션	코멘트
int	OnCalculate	const int rates_total, const int prev_calculated, const datetime &Time[], const double &Open[], const double &High[], const double &Low[], const double &Close[], const long &TickVolume[], const long &Volume[], const int &Spread[]	지표	모든 가격에 대한 이벤트 핸들러를 계산.
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	지표	계산: 단순 데이터 배열에서 이벤트 핸들러를. 지표에는 두 개의 이벤트 핸들러를 동시에 가질 수 없습니다. 이 경우 유일한 이벤트 핸들러를 데이터 배열로 사용할 것입니다.
void	OnTick	none	Expert Advisor	NewTick 이벤트 핸들러. 새 틱 수신 이벤트가 처리되는 동안 이 유형의 다른 이벤트는 수신되지 않습니다.
void	OnTimer	none	Expert Advisor 및 지표	타이머 이벤트 핸들러.
void	OnTrade	none	Expert Advisor	트레이드 이벤트 핸들러.
double	OnTester	none	Expert Advisor	테스터 이벤트 핸들러.
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	Expert Advisor 및 지표	ChartEvent 이벤트 핸들러.
void	OnBookEvent	const string &symbol_name	Expert Advisor 및 지표	BookEvent 이벤트 핸들러.

클라이언트 터미널은 해당하는 오픈 차트에 새 이벤트를 보냅니다. 차트(chart events) 또는 mql5-programs (커스텀 이벤트)에서 이벤트를 생성할 수도 있습니다. CHART_EVENT_OBJECT_CREATE 및 CHART_EVENT_OBJECT_DELETE 차트 속성을 설정하여 차트에 그래픽 개체를 생성 또는 삭제하는 이벤트

© 2000-2025, MetaQuotes Ltd.

를 생성하거나 생성할 수 있습니다. 각 MQL5 프로그램과 각 차트에는 모든 새 수신 이벤트가 추가되는 고유한 이벤트 대기열이 있습니다.

프로그램은 실행되는 차트에서만 이벤트를 수신합니다. 모든 이벤트가 수신되는 순서대로 차례로 처리됩니다. 대기열에 이미 NewTick 이벤트가 있거나 이 이벤트가 현재 처리 중인 경우 새 NewTick 이벤트가 MQL5 프로그램의 대기열에 배치되지 않습니다. 마찬가지로, ChartEvent가 이미 대기열에 있거나 이 이벤트가 처리 중인 경우 이러한 종류의 새 이벤트는 대기열에 포함되지 않습니다. 타이머 이벤트는 동일한 방식으로 처리됩니다. Timer 이벤트가 대기열에 있거나 처리 중인 경우 새 타이머 이벤트가 대기열에 있지 않습니다.

이벤트 대기열의 크기는 제한적이지만 충분하므로 제대로 작성된 프로그램을 위한 대기열 오버플로가 발생하지 않습니다. 대기열 오버플로의 경우 새 이벤트가 대기열 없이 삭제됩니다.

이벤트를 처리하기 위해 무한 루프를 사용하지 않는 것이 좋습니다. 단일 Start 이벤트를 처리하는 스크립트 및 서비스가 예외일 수 있습니다.

Libraries는 어떤 이벤트도 처리하지 않습니다.

지표 및 Expert Advisor에서 금지되는 함수

지표, 스크립트 및 Expert Advisor는 MQL5로 작성된 실행 가능한 프로그램입니다. 그것들은 다양한 종류의 작업을 위해 설계되었습니다. 따라서 프로그램 종류에 따라 특정 기능의 사용에 대한 몇 가지 제한이 있습니다. 지표에서는 다음 기능이 금지됩니다:

- OrderCalcMargin();
- OrderCalcProfit();
- OrderCheck();
- OrderSend();
- SendFTP();
- Sleep();
- ExpertRemove();
- MessageBox().

지표용으로 설계된 모든 함수는 Expert Advisor 및 스크립트에서 금지됩니다:

- SetIndexBuffer();
- IndicatorSetDouble();
- IndicatorSetInteger();
- IndicatorSetString();
- PlotIndexSetDouble();
- PlotIndexSetInteger();
- PlotIndexSetString();
- PlotIndexGetInteger.

© 2000-2025, MetaQuotes Ltd.

라이브러리는 독립적인 프로그램이 아니며 스크립트, 지표 또는 Expert Advisor라고 부르는 MQL5 프로그램의 컨텍스트에서 실행됩니다. 따라서 위의 제한 사항은 호출된 라이브러리에 적용됩니다.

서비스에서 금지된 함수

서비스는 어떤 이벤트도 차트에 바인딩되어 있지 않으므로 수락하지 않습니다. 서비스에서는 다음 기능을 사용할 수 없습니다:

ExpertRemove();
EventSetMillisecondTimer();
EventSetTimer();
EventKillTimer();
SetIndexBuffer();
IndicatorSetDouble();
IndicatorSetInteger();
IndicatorSetString();
PlotIndexSetDouble();
PlotIndexSetInteger();
PlotIndexSetString();
PlotIndexGetInteger();

지표의 로딩 및 언로딩

지표는 다음과 같은 경우에 로드됩니다:

- 지표가 차트에 첨부됨;
- 터미널 시작(터미널 정지 전에 지표가 차트에 부착된 경우);
- 템플릿 로드(차트에 첨부된 지표가 템플릿에 지정된 경우);
- 프로필 변경(지표가 프로필 차트 중 하나에 연결된 경우);
- 지표가 첨부된 차트의 심볼 및/또는 타임프레임 변경;
- 터미널이 연결된 계정의 변경;
- 지표의 재작성에 성공한 이후(지표가 차트에 부착된 경우);
- 지표의 입력 파라미터 변경.

다음과 같은 경우 지표가 언로드됩니다:

- 차트에서 지표를 분리할 때;
- 터미널 정지(지표가 차트에 첨부된 경우);
- 템플릿 로드(지표가 차트에 연결된 경우);
- 지표가 첨부된 차트의 마감;

© 2000-2025, MetaQuotes Ltd.

- 프로필 변경(지표가 변경된 프로필의 차트 중 하나에 연결된 경우);
- 지표가 첨부된 차트의 심볼 및/또는 타임프레임 변경;
- 터미널이 연결된 계정의 변경;
- 지표의 입력 파라미터 변경.

Expert Advisor의 로딩 및 언로딩

Expert Advisor는 다음과 같은 경우에 로드됩니다:

- 차트에 Expert Advisor를 첨부할 때;
- 터미널 시작(터미널 종료 전에 Expert Advisor가 차트에 첨부된 경우);
- 템플릿 로드(차트에 첨부된 Expert Advisor가 템플릿에 지정된 경우);
- 프로파일 변경(Expert Advisor가 프로파일 차트 중 하나에 연결된 경우);
- 계정 번호가 동일한 경우임에도 계정에 대한 연결(서버의 터미널 승인 전에 Expert Advisor를 차트에 첨부한 경우).

다음과 같은 경우 Expert Advisors는 언로드됩니다:

- 차트에서 Expert Advisor를 분리할 때;
- 차트에 새 Expert Advisor가 첨부되었는데 다른 Expert Advisor가 이미 첨부되어 있는 경우, 해당 Expert Advisor는 언로드됩니다.
- 터미널 종료(Expert Advisor를 차트에 첨부한 경우);
- 템플릿 로드(Expert Advisor가 차트에 첨부된 경우);
- Expert Advisor가 첨부된 차트 마감.
- 프로필 변경(Expert Advisor가 변경된 프로필의 차트 중 하나에 첨부된 경우);
- 터미널이 연결된 계정의 변경(서버의 터미널 승인 전에 Expert Advisor를 차트에 첨부한 경우);
- ExpertRemove() 함수를 호출.

Expert Advisor가 연결된 차트의 심볼이나 타임프레임이 변경되는 경우 Expert Advisor는 로드되거나 언로드되지 않습니다. 이 경우 클라이언트 터미널은 이후 이전 심볼/타임프레임에서 OnDeinit() 핸들러를 호출하고 새 심볼/타임프레임에서 OnInit()를 호출하며, 글로벌 변수와 정적 변수의 값은 재설정되지 않습니다. 초기화가 완료되기 전에 Expert Advisor에 수신된 모든 이벤트 (OnInit() 함수)는 건너뜁니다.

스크립트의 로딩 및 언로딩

스크립트는 차트에 연결된 후 즉시 로드되고 작업을 완료한 후 즉시 언로드됩니다. OnInit() 및 OnDeinit()는 스크립트에서는 호출되지 않습니다.

프로그램이 언로드되면(차트에서 삭제됨) 클라이언트 터미널은 글로벌 변수의 초기화를 수행하고 이벤트 대기열을 삭제합니다. 이 경우 초기화 해제는 모든 string-유형 변수의 재설정, 동적 배열 객체의 할당 해제 및 소멸자 호출을 의미합니다.

서비스의 로딩 및 언로딩

© 2000-2025, MetaQuotes Ltd.

터미널 종료 시점에 서비스를 시작한 경우 터미널 시작 후 바로 서비스가 로드됩니다. 서비스는 작업을 마친 후 즉시 언로드됩니다.

서비스에는 네트워크 기능을 사용해 커스텀 심볼을 만들고 업데이트하는 등 데이터 수신 및 처리 루프를 무한대로 구현할 수 있는 단일 OnStart() 핸들러가 있습니다.

지표와 스크립트, Expert Advisor와는 달리 서비스는 특정 차트에 구속되지 않으므로 서비스를 시작할 수 있는 별도의 메커니즘이 제공됩니다. 새 서비스 인스턴스는 "서비스 추가" 명령을 사용하여 네비게이터에 작성됩니다. 적절한 인스턴스 메뉴를 사용하여 서비스 인스턴스를 시작, 중지 및 제거할 수 있습니다. 모든 인스턴스를 관리하기 위해 서비스 메뉴를 사용하십시오.

Expert Advisor 작업에 대한 이해를 높이기 위해 다음 Expert Advisor의 코드를 컴파일하고 로드/언로드, 템플릿 변경, 심볼 변경, 타임프레임 변경 등의 작업을 수행하는 것이 좋습니다.

예:

```
//+------------------------------------------------------------------+
//|                                                   TestExpert.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

class CTestClass
  {
public:
   CTestClass() { Print("CTestClass constructor"); }
   ~CTestClass() { Print("CTestClass destructor"); }
  };
CTestClass global;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Initialization");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 취소 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
```

© 2000-2025, MetaQuotes Ltd.

```
    Print("근거와 함께 초기화 취소",reason);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
```

참고 항목

Client terminal events, Event handlers

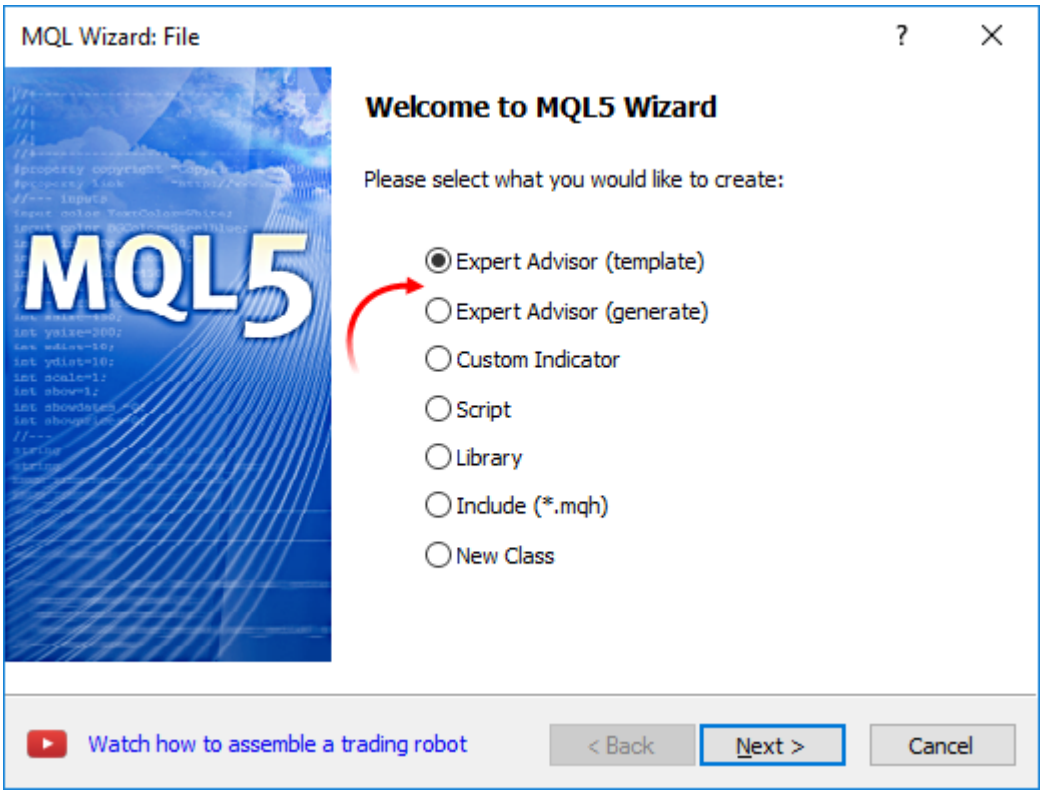
© 2000-2025, MetaQuotes Ltd.

트레이드 퍼미션

트레이드 오토메이션

MQL5 언어는 자동화된 거래 시스템 개발을 위해 설계된 거래 함수의 특별한 그룹을 제공합니다. 인간의 개입 없이 자동화된 거래를 위해 개발된 프로그램을 Expert Advisors 또는 Trade Robots라고 부릅니다. MetaEditor에서 Expert Advisor를 생성하려면 MQL5 Wizard를 시작하고 다음 두 가지 옵션 중 하나를 선택합니다:

- Expert Advisor (템플릿) – 를 사용하면 프로그래밍을 통해 필요한 모든 함수로 보완되어야 하는 미리 작성된 이벤트 핸들링 함수가 있는 템플릿을 만들 수 있습니다.
- Expert Advisor (생성) – 완전한 트레이딩 로봇을 개발할 수 있도록 함 필요한 모듈(거래 신호 모듈, 머니 관리 모듈 및 후행 정지 모듈)을 선택하면 됩니다.



거래 함수는 Expert Advisor 및 스크립트에서만 사용할 수 있습니다. 지표에는 거래가 허용되지 않습니다.

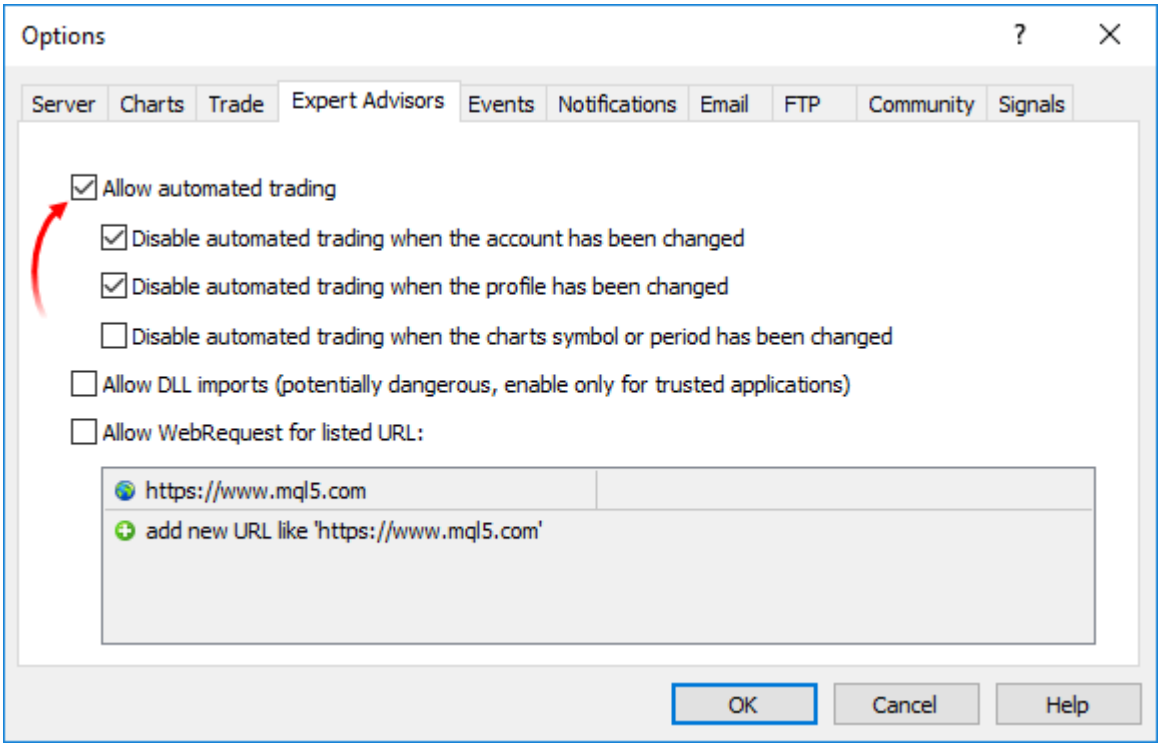
자동 거래 수행 권한 확인

사람의 개입 없이 작업할 수 있는 신뢰할 수 있는 Expert Advisor을 개발하기 위해서는 일련의 중요한 검사를 준비할 필요가 있습니다. 우선, 우리는 프로그램적으로 거래가 허용되는지 확인해야 합니다. 이것은 자동화 시스템을 개발할 때 필수적인 기본 점검입니다.

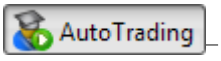
터미널에서 자동 거래를 수행할 수 있는 권한을 확인하는 중

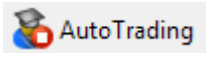
터미널 설정을 사용하면 모든 프로그램에 대해 자동 거래를 허용하거나 금지할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.



터미널의 Standard 패널에서 바로 자동 거래 옵션을 전환할 수 있습니다:

- 

– 자동화 거래를 출시했으며 애플리케이션의 거래 함수 사용 허용됨.
- 

– 자동 거래가 비활성화되어 실행 중인 어플리케이션이 거래 함수를 실행할 수 없습니다.

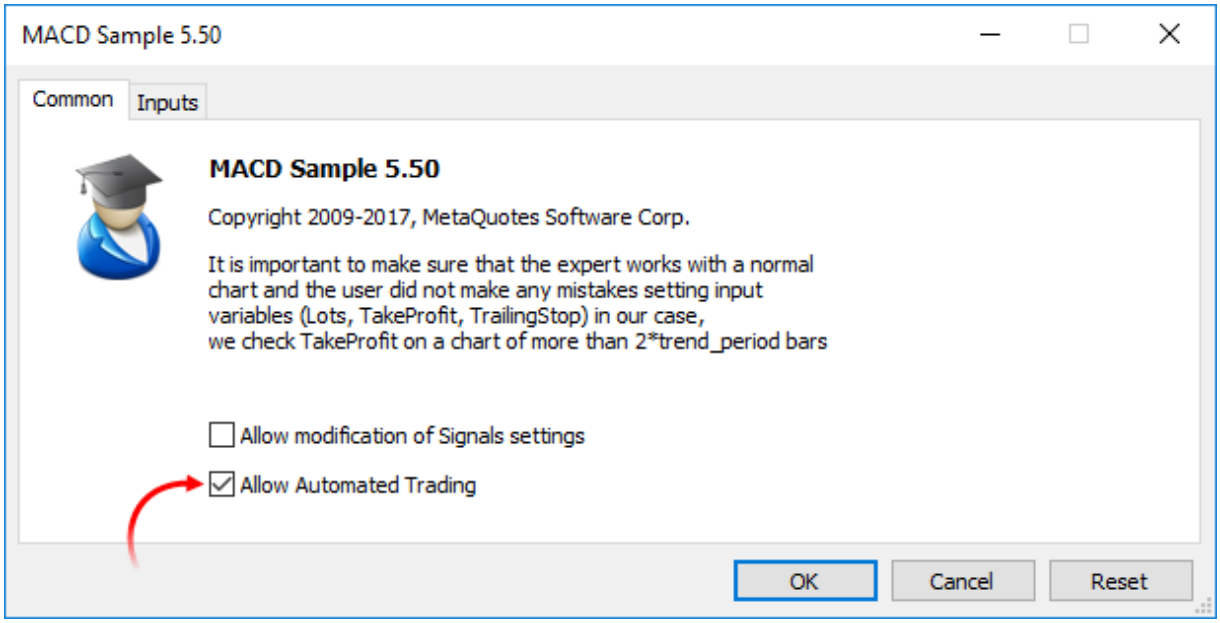
샘플 체크:

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("터미널 설정에서 자동 거래가 가능한지 확인하십시오!");
```

실행 중인 특정 Expert Advisor/script에 대해 거래가 허용되는지 확인

또는 금지할 때 개시한 특정 프로그램을 위한 자동화된 거래 허용할 수 있습니다. 이렇게 하려면 프로그램 속성에서 특수 확인란을 사용합니다.

© 2000-2025, MetaQuotes Ltd.



샘플 체크:

```
if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("터미널 설정에서 자동 거래가 가능한지 확인하십시오!");
else
  {
   if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
      Alert("다음의 프로그램 설정에는 자동 거래가 금지됩니다: ",__FILE__);
  }
```

현재 계정에 대해 Expert Advisor/scripts 거래가 허용되는지 확인

거래 서버 측에서 자동 거래를 비활성화할 수 있습니다. 샘플 체크:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
   Alert("계정에 대해 자동 거래가 금지됩니다. ",AccountInfoInteger(ACCOUNT_LOGIN),
   " 거래 서버 측에서");
```

거래 계정에 대해 자동 거래가 비활성화된 경우, Expert Advisor/scripts의 거래 작업이 실행되지 않습니다.

현재 계정에 대해 거래가 허용되는지 확인

경우에 따라 특정 거래 계좌에 대해 모든 거래 업무가 비활성화되며, 수동 거래나 자동 거래를 수행할 수 없습니다. 거래처 연결에 투자자 암호를 사용한 경우 샘플 체크:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
   Comment("계정에 대한 거래가 금지되어 있습니다. ",AccountInfoInteger(ACCOUNT_LOGIN),
         ".₩n 거래 계정에 연결하기 위해 투자자 비밀번호가 사용되었을 수 있습니다.",
         "₩n 터미널 저널에서 다음 항목을 확인하십시오:",
         "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"₩: 거래가 비활성화되었습니다 - 투자
```

AccountInfoInteger(ACCOUNT_TRADE_ALLOWED)는 다음과 같은 경우 **false**를 반환할 수 있습니다:

© 2000-2025, MetaQuotes Ltd.

- 거래 서버에 연결되지 않았습니다. TerminalInfoInteger(TERMINAL_CONNECTED)를 사용하여 확인할 수 있습니다;
- 거래 계정이 읽기 전용 모드로 전환됨(아카이브로 전송됨);
- 거래 서버 측에서 계정에 대한 거래가 비활성화되어 있습니다.
- 거래 계정에 대한 연결이 투자자 모드에서 수행되었습니다.

참고 항목

Client Terminal Properties, Account Properties, Properties of a Running MQL5 Program

© 2000-2025, MetaQuotes Ltd.

클라이언트 터미널 이벤트

Init

클라이언트 터미널이 프로그램(Expert Advisor 또는 커스텀 지표)을 로드하고 글로벌 변수 초기화 프로세스를 시작하면 곧바로 Init 이벤트가 전송되며, Init 이벤트는 OnInit() 이벤트 핸들러가 처리합니다. 이 이벤트는 또한 금융상품 및/또는 차트 시간 변경 후, 프로그램이 MetaEditor에서 다시 컴파일된 후, Expert Advisor 또는 커스텀 지표의 설정 창에서 입력 매개 변수가 변경된 후에 생성됩니다. 계정 변경 후 Expert Advisor도 초기화됩니다. 스크립트에 대해 Init 이벤트가 생성되지 않습니다.

Deinit

글로벌 변수가 초기화 해제되고 프로그램(Expert Advisor 또는 커스텀 지표)이 언로드되기 전에 클라이언트 터미널은 Deinit 이벤트를 프로그램으로 전송합니다. 클라이언트 터미널이 닫혔을 때, 차트가 닫혔을 때, 보안 및/또는 기간이 변경되기 직전에, 성공적인 프로그램 재컴파일에서, 입력 매개 변수가 변경되었을 때, 계정 변경 시에도 Deinit가 생성됩니다.

The 초기화 취소 사유는 매개 변수에서 얻을 수 있으며, OnDeinit() 함수에 전달됩니다. OnDeinit() 함수 실행은 2.5초로 제한됩니다. 이 시간 동안 기능이 완료되지 않으면 강제로 종료됩니다. 스크립트에 대해 Deinit 이벤트가 생성되지 않습니다.

Start

Start는 스크립트 또는 서비스를 로드한 후 시작하는 특별한 이벤트입니다. 이것은 OnStart 함수로 제어됩니다. 시작 이벤트는 EA 및 커스텀 지표에 전달되지 않습니다.

NewTick

새로운 견적이 있을 경우 NewTick 이벤트가 생성되며, 첨부된 Expert Advisor의 OnTick()에 의해 처리됩니다. 새 견적을 받을 때 이전 견적서의 OnTick 함수가 처리되는 경우 해당 이벤트가 계속 진행되지 않기 때문에 Expert Advisor가 새 견적을 무시합니다.

프로그램이 실행되는 동안 수신되는 모든 새 따옴표는 OnTick()이 완료될 때까지 무시됩니다. 그런 다음 새 견적을 받은 후에만 기능이 실행됩니다. NewTick 이벤트는 자동 거래가 허용되는지 여부에 관계없이 생성됩니다 ("자동 거래 허용/금지" 버튼). 자동 거래 금지는 Expert Advisor가 계속 작업하는 동안 Expert Advisor의 거래 요청을 보낼 수 없다는 것을 의미합니다.

적절한 버튼을 눌러 자동 거래를 금지해도 OnTick() 함수의 현재 실행이 중지되지는 않습니다.

Calculate

Calculate 이벤트는 Init 이벤트가 전송된 직후 및 가격 데이터의 변경 시 지표에 대해서만 생성됩니다. 이것은 OnCalculate 함수로 처리됩니다.

Timer

타이머 이벤트는 EventSetTimer 함수로로 타이머를 활성화한 Expert Advisor의 클라이언트 터미널에서 주기적으로 생성됩니다. 일반적으로 이 기능은 OnInit에 의해 호출됩니다. 타이머 이벤트 처리는 OnTimer X 기능에 의해 수행됩니다. Expert Advisor의 작동이 완료된 후에는 일반적으로 OnDeinit 함수에서 호출되는 EventKillTimer 함수를 사용하여 타이머를 파괴할 필요가 있습니다.

© 2000-2025, MetaQuotes Ltd.

Trade

트레이드 이벤트는 트레이드 서버에서 트레이드 작업이 완료되면 생성됩니다. 트레이드 이벤트는 다음 거래 작업들에 대해 OnTrade() 함수로 처리됩니다.

- 보류 중인 주문의 전송, 수정 또는 제거;
- 대금 또는 유효기간이 충분하지 않은 보류 주문의 취소;
- 보류 중인 주문의 활성화;
- 포지션(또는 그 일부) 오프닝, 추가, 또는 클로징
- 오픈 포지션의 수정 (스탑 변경 – 스탑 로스 및/또는 이익 획득).

TradeTransaction

거래 계정에 대해 몇 가지 확실한 작업을 수행할 때 거래 상태가 변경됩니다. 이러한 조치는 다음을 포함합니다.

- OrderSend 및 OrderSendAsync 함수를 사용하여 클라이언트 터미널의 모든 MQL5 응용프로그램에서 거래 요청 전송 및 추가 실행;
- 터미널 그래픽 인터페이스를 통해 거래 요청 전송 및 추가 실행;
- 서버에서 보류 중인 주문 및 주문 활성화 중지;
- 거래 서버 측에서 작업을 수행.

이러한 조치의 결과로 다음과 같은 트레이드 트랜잭션이 수행됩니다.

- 거래 요청 처리;
- 오픈 오더 변경;
- 주문 내역 변경;
- 거래 내역 변경;
- 포지션 변경.

예를 들어, 시장 매수 주문을 보낼 때, 그것은 처리되고, 계정에 대한 적절한 매수 주문이 생성되고, 그 주문이 실행되어 오픈 매수 주문 목록에서 제거되고, 그런 다음, 주문 내역에 적절한 거래가 추가되고, 새로운 포지션이 생성됩니다. 이 모든 행동들은 트레이드 트랜잭션입니다. 이러한 거래가 터미널에 도착하는 것은 TradeTransaction 이벤트입니다. 이 이벤트는 OnTradeTransaction 함수로 처리됩니다.

Tester

내역 정보에 대한 Expert Advisor 테스트가 끝나면 Tester 이벤트가 생성됩니다. 이벤트는 OnTester() 함수로 처리됩니다.

TesterInit

TesterInit 이벤트는 첫 번째 최적화가 통과하기 전에 전략 테스터에서 최적화의 시작과 함께 생성됩니다. TesterInit 이벤트는 OnTesterInit() 함수로 처리됩니다.

TesterPass

© 2000-2025, MetaQuotes Ltd.

TesterPass 이벤트는 새 데이터 프레임을 수신할 때 생성됩니다. TesterPass 이벤트는 OnTesterPass() 함수로 처리됩니다.

TesterDeinit

TesterDeinit 전략 테스터에서 Expert Advisor의 최적화가 종료된 후에 생성됩니다. TesterDeinit 이벤트는 OnTesterDeinit() 함수로 처리됩니다.

ChartEvent

ChartEvent 이벤트 생성은 사용자가 차트로 작업할 때 클라이언트 터미널에서 가능합다.

- 키 스트로크, 차트 창이 포커스에 있을 때;
- 그래픽 개체가 생성됐을 때;
- 그래픽 개체가 삭제됐을 때
- 차트의 그래픽 개체를 마우스로 클릭
- 마우스를 사용한 그래픽 개체 이동
- LabelEdit에서 텍스트 편집의 끝.

또한 EventChartCustom 함수를 사용하여 모든 mql5 프로그램에 의해 Expert Advisor에게 전송될 수 있는 커스텀 이벤트 ChartEvent도 있습니다. 이 이벤트는 OnChartEvent 함수로 처리됩니다.

BookEvent

BookEvent 이벤트는 Depth Of Market 변경 후 클라이언트 터미널에서 생성되며, OnBookEvent 함수에 의해 처리됩니다. 지정된 심볼에 대해 BookEvent 생성을 시작하려면 MarketBookAdd 함수를 사용하여 이 이벤트에 심볼을 구독해야 합니다.

지정된 심볼에 대한 BookEvent 구독을 취소하려면 MarketBookRelease 함수를 호출해야 합니다. BookEvent 이벤트는 방송형 이벤트로, 이 이벤트에 대해 Expert Advisor 1명만 가입해도 충분하며, OnBookEvent 이벤트 핸들러가 있는 다른 모든 Expert Advisor도 이를 수신합니다. 이러한 이유로 핸들러에 매개 변수로 전달되는 심볼 이름을 분석해야 합니다.

참고 항목

Event handlers, Program running

© 2000-2025, MetaQuotes Ltd.

리소스

MQL5 프로그램에서 그래픽 및 사운드 사용

MQL5의 프로그램을 사용하면 사운드 및 그래픽 파일 작업을 수행할 수 있습니다:

- PlaySound()는 사운드 파일을 플레이합니다;
- ObjectCreate()를 사용하면 그래픽 개체 OBJ_BITMAP 및 OBJ_BITMAP_LABEL 을 사용하여 유저 인터페이스를 만들 수 있습니다.

PlaySound()

PlaySound() 함수의 호출 예:

```
//+------------------------------------------------------------------+
//| 표준 OrderSend()를 호출하고 사운드를 재생합니다                  |
//+------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
  //--- 서버에 요청 전송
   OrderSend(request,result);
   //--- 요청이 수락되면 사운드 Ok.wav를 재생합니다.
   if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
   //--- 실패할 경우, timeout.wav 파일에서 알람을 재생합니다
   else PlaySound("timeout.wav");
  }
```

이 예에서는 표준 터미널 패키지에 포함되어 있는 'OK.wav' 및 'timeout' 파일에서 소리를 재생하는 방법을 보여 줍니다. 이러한 파일은 폴더 **terminal_directory\Sounds**에 있습니다. 여기서 **terminal_directory**는 MetaTrader 5 클라이언트 터미널이 시작되는 폴더입니다. 터미널 디렉터리의 위치는 다음과 같은 방법으로 mql5 프로그램에서 확인할 수 있습니다:

```
//--- 터미널 데이터가 저장되는 폴더
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

terminal_directory\Sounds폴더뿐만 아니라 다음 폴더에 있는 모든 하위 폴더에서도 사운드 파일을 사용할 수 있습니다 : **terminal_data_directory\MQL5.** 터미널 메뉴 "File" -> "Open Data Folder"에서 터미널 데이터 디렉토리의 위치를 확인하거나 프로그램 방법을 사용하여 확인할 수 있습니다:

```
//--- 터미널 데이터가 저장되는 폴더
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

예를 들어, Demo.wav 사운드 파일이 terminal_data_directory₩MQL5₩Files 에 있는 경우, PlaySound()의 호출은 다음과 같이 작성되어야 합니다:

```
//--- terminal_directory_data₩MQL5₩Files₩ 폴더에서 Demo.wav 재생
   PlaySound("\\Files\\Demo.wav");
```

주석에서 파일의 경로는 백슬래시 "₩, 함수 "₩₩를 사용하여 작성된다는 점에 유의하십시오.

© 2000-2025, MetaQuotes Ltd.

경로를 지정할 때는 항상 이중 백슬래시만 구분 기호로 사용하십시오. 단일 백슬래시는 프로그램 소스 코드에서 상수 문자열과 문자 상수를 처리할 때 컴파일러의 제어 기호이기 때문입니다.

NULL 매개 변수로 PlaySound()함수를 호출하여 재생을 중지합니다:

```
//--- NULL 매개 변수를 사용하여 PlaySound()를 호출하면 재생이 중지됩니다
   PlaySound(NULL);
```

ObjectCreate()

ObjectCreate() 함수를 사용하는 그래픽 레이블(OBJ_BITMAP_LABEL)을 생성하는 Expert Advisor의 예.

```
string label_name="currency_label";         // name of the OBJ_BITMAP_LABEL object
string euro      ="\\Images\\euro.bmp";     // path to the file terminal_data_directory
string dollar    ="\\Images\\dollar.bmp";   // path to the file terminal_data_directory
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 아직 OBJ_BITMAP_LABEL 버튼이 생성되지 않았다면, 이를 생성합니다
   if(ObjectFind(0,label_name)<0)
     {
      //--- OBJ_BITMAP_LABEL 개체를 생성 시도
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- 버튼을 차트의 왼쪽 상단 코너에 링크
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
         //--- 개체 속성을 설정
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- 마지막 오류의 코드를 0으로 재설정
         ResetLastError();
         //--- 사진을 다운로드하여 버튼의 "누름" 상태를 표시
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- 결과 테스트
         if(!set)
           {
            PrintFormat("%s 파일에서 이미지 다운로드 실패. Error code %d",euro,GetLastErr
           }
         ResetLastError();
         //--- 사진을 다운로드하여 버튼의 "누르지 않음" 상태를 표시
         set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

         if(!set)
           {
            PrintFormat("%s 파일에서 이미지 다운로드 실패. Error code %d",dollar,GetLastE
```

© 2000-2025, MetaQuotes Ltd.

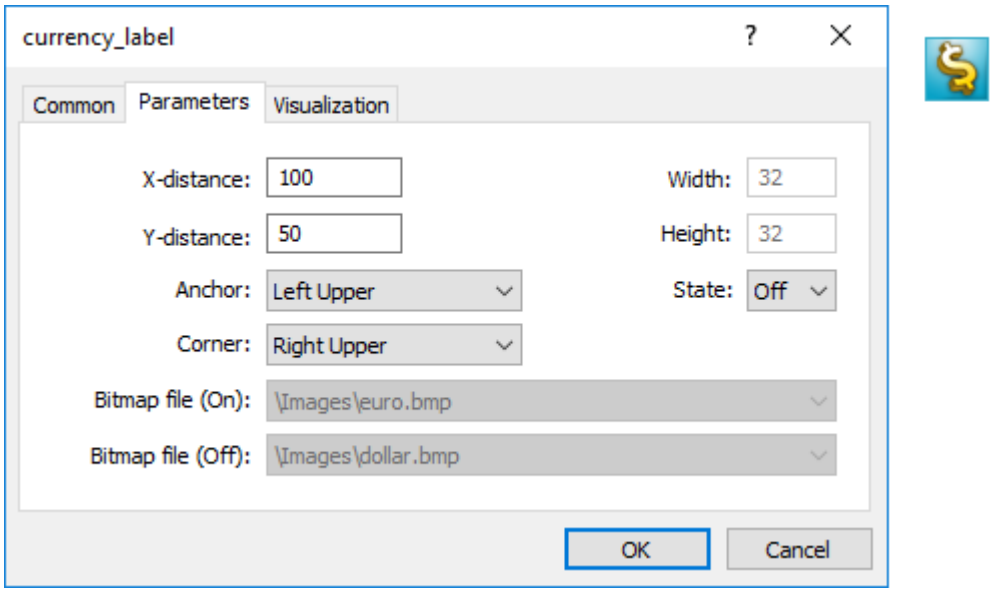
```
            }
          //--- 차트 새로 고침 명령을 전송하여 버튼이 즉시 나타남
          ChartRedraw(0);
         }
       else
         {
          //--- 개체 생성 실패, 알림
          PrintFormat("OBJ_BITMAP_LABEL 오브젝트 생성 실패. Error code %d",GetLastError()
         }
      }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 취소 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 차트에서 객체를 삭제
   ObjectDelete(0,label_name);
  }
```

currency_label이라는 그래픽 객체의 생성 및 설정은 다음 함수에서 수행됩니다: OnInit() function. 그래픽 파일의 경로는 글로벌 변수 유로 및 달러로 설정되며, 구분 기호에 이중 백슬래시가 사용됩니다:

```
string euro      ="\\Images\\euro.bmp";    // path to the file terminal_dara_directory
string dollar    ="\\Images\\dollar.bmp";  // path to the file terminal_dara_directory
```

파일은 다음 폴더에 위치합니다 **terminal_data_directory\MQL5\Images.**

OBJ_BITMAP_LABEL 개체는 실제로 버튼 상태(누름 또는 누르지않음)에 따라 두 이미지 중 하나를 표시하는 버튼입니다: euro.bmp 또는 dollar.bmp.



© 2000-2025, MetaQuotes Ltd.

그래픽 인터페이스가 있는 버튼의 크기는 그림 사이즈에 맞게 자동으로 조정됩니다. 이미지는 OBJ_BITMAP_LABEL 개체를 마우스 좌클릭하면 변경됩니다 ("**선택 사용 안 함**" **옵션은 속성에서 선택해야 함**). OBJ_BITMAP 개체는 동일한 방식으로 생성되며, 필요한 이미지로 배경을 만드는 데 사용됩니다.

다음 개체의 모양을 담당하는 OBJPROP_BMPFILE 속성 값은 동적으로 변경할 수 있습니다: OBJ_BITMAP 및 OBJ_BITMAP_LABEL. 이를 통해 mql5 프로그램에 대한 다양한 대화형 사용자 인터페이스를 만들 수 있습니다.

mql5 프로그램을 컴파일하는 동안 실행 파일에 리소스 포함

mql5 프로그램은 이미지와 사운드 파일의 형태로 많은 다양한 다운로드 가능한 자원이 필요할 수 있습니다. MQL5에서 실행 파일을 이동할 때 이러한 파일을 모두 전송할 필요가 없도록 컴파일러의 명령문 #resource를 사용해야 합니다:

```
#resource path_to_resource_file
```

#resource 명령은 컴파일러에게 지정된 경로 **path_to_resource_file**의 리소스를 실행 가능한 EX5 파일에 포함해야 한다고 알려줍니다. 따라서 필요한 모든 영상과 사운드는 EX5 파일에 직접 위치할 수 있으므로, 프로그램을 다른 단말에서 실행하려면 EX5 파일에 사용되는 파일을 별도로 전송할 필요가 없습니다. 모든 EX5 파일은 리소스를 포함할 수 있으며, 모든 EX5 프로그램은 다른 EX5 프로그램의 리소스를 사용할 수 있습니다.

BMP 및 WAV 형식의 파일은 EX5 파일에 포함되기 전에 자동으로 압축됩니다. 이것은 MQL5에서 완전한 프로그램을 만들 뿐만 아니라, 자원을 사용하는 것은 MQL5 프로그램 쓰기의 일반적인 방법에 비해 그래픽과 사운드를 사용할 때 필요한 파일의 전체 크기를 줄일 수 있다는 것을 의미합니다.

리소스 파일 크기는 128Mb를 초과할 수 없습니다.

컴파일러에서 지정된 리소스 검색

#resource "<path to the resource file>" 명령을 사용하여 리소스를 삽입합니다.

```
#resource "<path_to_resource_file>"
```

상수 문자열 <path_to_resource_file>의 길이는 63자를 초과할 수 없습니다.

컴파일러는 다음 순서로 지정된 경로에서 리소스를 검색합니다:

- 백슬래시 "₩구분자("₩₩로 쓰임)가 경로의 시작 부분에 배치되어 있는 경우 다음 디렉토리에 상대적 자원을 검색합니다: **terminal_data_directory\MQL5**,
- 백슬래시가 없는 경우 리소스가 기록되는 소스 파일의 위치를 기준으로 리소스를 검색합니다.

리소스 경로는 하위 문자열 "..₩₩ and ":₩₩ 입니다.

리소스 포함 예:

```
//--- 자원의 정확한 사양
#resource "\\Images\\euro.bmp" // euro.bmp is located in terminal_data_directory\MQL5\
#resource "picture.bmp"        // picture.bmp는 원본 파일과 동일한 디렉토리에 있습니다.
```

© 2000-2025, MetaQuotes Ltd.

```
#resource "Resource\\map.bmp"  // the resource is located in source_file_directory\Res

//--- 리소스의 부정확한 사양
#resource ":picture_2.bmp"     // must not contain ":"
#resource "..\\picture_3.bmp"  // must not contain ".."
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" //63자
```

리소스 사용

리소스명

#resource 명령문을 사용하여 리소스를 선언한 후에는 프로그램의 모든 부분에서 사용할 수 있습니다. 리소스 이름은 줄 시작 부분에 백슬래시가 없는 경로로, 리소스 경로를 설정합니다. 코드에서 사용자 자신의 리소스를 사용하려면 리소스 이름 앞에 특수 기호 "::"를 추가해야 합니다.

예:

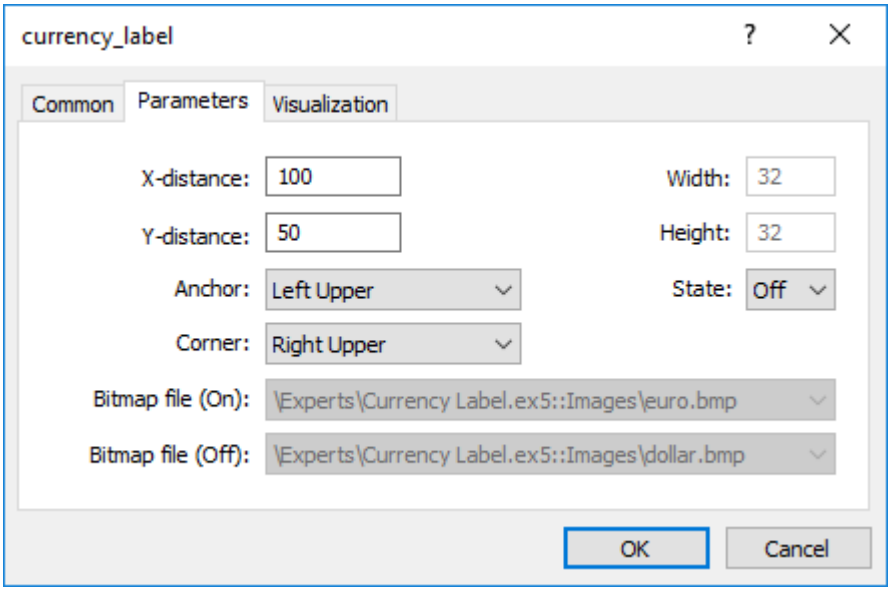
```
//--- 리소스 규격의 예 및 주석의 이름
#resource "\\Images\\euro.bmp"           // 리소스명 - Images\euro.bmp
#resource "picture.bmp"                  // 리소스명 - picture.bmp
#resource "Resource\\map.bmp"            // 리소스명 - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // 리소스명 - Files\Pictures\good.bmp
#resource "\\Files\\Demo.wav";           // 리소스명 - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";        // 리소스명 - Sounds\thrill.wav"
...

//--- 자원 활용
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

리소스의 이미지를 OBJ_BITMAP 및 OBJ_BITMAP_LABEL 설정할 때는 OBJPROP_BMPFILE 속성 값을 수동으로 수정할 수 없습니다. 예를 들어 OBJ_BITMAP_LABEL 를 생성하는 경우 리소스 euro.bmp 및 dollar.bmp를 이용합니다.

```
#resource "\\Images\\euro.bmp";    // euro.bmp is located in terminal_data_directory\M
#resource "\\Images\\dollar.bmp";  // dollar.bmp is located in terminal_data_directory
```

이 개체의 속성을 조회할 때 BitMap File (On) 및 BitMap File (Off) 속성이 흐리게 표시되고 수동으로 변경할 수 없습니다.:

© 2000-2025, MetaQuotes Ltd.



다른 mql5 프로그램의 리소스 사용

리소스를 사용하는 또 다른 이점이 있습니다 – 모든 MQL5 프로그램에서는 다른 EX5 파일의 리소스를 사용할 수 있습니다. 따라서 한 EX5 파일의 리소스를 다른 많은 mql5 프로그램에서 사용할 수 있습니다.

다른 파일의 리소스 이름을 사용하려면 <path_EX5_file_name>::<resource_name>로 지정해야 합니다. 예를 들어 Draw_Triangles_Script.mq5 스크립트에 triangle.bmp 파일의 이미지에 대한 리소스가 포함되어 있다고 가정합니다.

```
#resource "\\Files\\triangle.bmp"
```

그런 다음 스크립트 자체에서 사용하기 위한 그것의 이름은 "Files₩triangle.bmp"처럼 보일 것이고, 그것을 사용하기 위해서는 "::"가 리소스 이름에 추가되어야 합니다.

```
//--- 스크립트의 리소스 사용
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

다른 프로그램(예: Expert Advisor)의 동일한 리소스를 사용하려면 리소스 이름에 **terminal_data_directory\MQL5**에 대한 EX5 파일의 경로와 스크립트의 EX5 파일 이름 **Draw_Triangles_Script.ex5**를 추가해야 합니다. 스크립트가 표준 폴더 **terminal_data_directory\MQL5**Scripts₩에 있다고 가정하면 호출은 다음과 같이 작성되어야 합니다:

```
//--- EA의 스크립트에서 리소스 사용
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

다른 EX5에서 리소스를 호출할 때 실행 파일의 경로를 지정하지 않으면 리소스를 호출하는 프로그램이 포함된 동일한 폴더에서 실행 파일이 검색됩니다. 즉, 경로 지정 없이 Expert Advisor가 Draw_Triangles_Script.ex5 에서 리소스를 호출할 경우 다음과 같이 됩니다:

```
//--- 경로를 지정하지 않고 EA의 스크립트 리소스 호출
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

terminal_data_directory\MQL5 Experts₩ 폴더에서 파일을 검색하십시오, Expert Advisor가 **terminal_data_directory\MQL5**Experts₩에 있다면.

© 2000-2025, MetaQuotes Ltd.

커스텀 지표가 리소스로 포함되도록 작업

MQL5 어플리케이션들의 작동을 위해 하나 또는 여러 개의 커스텀 지표가 필요할 수 있습니다. 이들 모두는 실행 가능한 MQL5 프로그램의 코드에 포함될 수 있습니다. 지표가 자원으로 포함되면 어플리케이션 배포가 간단해집니다.

다음은 **terminal_data_folder₩MQL5₩Indicators₩directory** 에 있는 SampleIndicator.ex5 커스텀 지표를 포함 및 사용하는 예입니다:

```
//+------------------------------------------------------------------+
//|                                                     SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
   if(handle_ind==INVALID_HANDLE)
     {
      Print("Expert: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
//--- ...
   return(INIT_SUCCEEDED);
  }
```

OnInit() 함수의 커스텀 지표가 하나 이상의 사본을 생성하는 경우 특별한 고려가 필요합니다. Please keep in mind that the resource should be specified in the following way리소스는 다음과 같은 방법으로 지정해야 합니다.**<path_EX5_file_name>::<resource_name>.**

예를 들어 SampleIndicator.ex5 지표가 SampleEA.ex5 Expert Advisor에 리소스로 포함된 경우 커스텀 지표의 초기화 함수에서 iCustom() 를 호출할 때 지정된 자체 경로는 다음과 같습니다: "₩Experts₩₩SampleEA.ex5::Indicators₩₩SampleIndicator.ex5". 이 경로를 명시적으로 설정하면 SampleIndicator.ex5 커스텀 지표는 독립적으로 작동하는 능력을 상실하는 SampleEA.ex5 Expert Advisor에 견고하게 연결됩니다.

자체 경로는 GetRelativeProgramPath() 함수를 사용하여 수신할 수 있습니다. 사용 예는 아래에 나와 있습니다:

```
//+------------------------------------------------------------------+
//|                                               SampleIndicator.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 스스로의 링크를 제공하는 잘못된 방법
//--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
//--- 스스로의 링크를 받는 올바른 방법
  string path=GetRelativeProgramPath();
//--- 지표 버퍼 매핑
   handle=iCustom(_Symbol,_Period,path,0,0);
   if(handle==INVALID_HANDLE)
     {
      Print("지표: iCustom call: Error code=",GetLastError());
      return(INIT_FAILED);
     }
   else Print("Indicator handle=",handle);
//---
   return(INIT_SUCCEEDED);
  }
///....
//+------------------------------------------------------------------+
//| GetRelativeProgramPath                                           |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
  {
   int pos2;
//--- 어플리케이션의 절대 경로 획득
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- "₩MQL5₩" 하위 문자열의 위치를 탐색
   int    pos =StringFind(path,"\\MQL5\\");
//--- 하위문자열 찾을 수 없음 - 오류
   if(pos<0)
      return(NULL);
//--- skip "\MQL5" directory
   pos+=5;
//--- skip extra '\' symbols
   while(StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- 리소스인 경우 MQL5 디렉토리를 기준으로 경로를 반환합니다.
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//--- 첫 번째 MQL5 하위 디렉토리에 대한 구분자 찾기(예: MQL5₩Indicators)
//--- 찾을 수 없는 경우 MQL5 디렉토리를 기준으로 경로를 반환합니다.
   if((pos2=StringFind(path,"\\",pos+1))<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      return(StringSubstr(path,pos));
//--- 하위 디렉토리에 상대 경로 반환 (예: MQL5\Indicators)
   return(StringSubstr(path,pos2+1));
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double& price[])
  {
//--- 다음 요청에 대한 prev_calculated 의 값 반환
   return(rates_total);
  }
```

리소스 변수

리소스 변수를 사용하여 리소스를 선언하고 적절한 유형의 변수인 것처럼 처리할 수 있습니다. 선언 형식 :

```
#resource path_to_the_resource_file as resource_variable_type resource_variable_name
```

선언 샘플:

```
#resource "data.bin" as int ExtData[]              // data.bin 파일 데이터를 포함하는 숫자
#resource "data.bin" as MqlRates ExtData[]         // data.bin 파일 데이터를 포함하는 단순
//--- strings
#resource "data.txt" as string ExtCode             // data.txt 파일 데이터가 포함된 문자열
//--- graphical resources
#resource "image.bmp" as bitmap ExtBitmap[]        // BMP 파일의 비트맵을 포함하는 1차원 배
#resource "image.bmp" as bitmap ExtBitmap2[][]     // BMP 파일의 비트맵을 포함하는 2차원 배
```

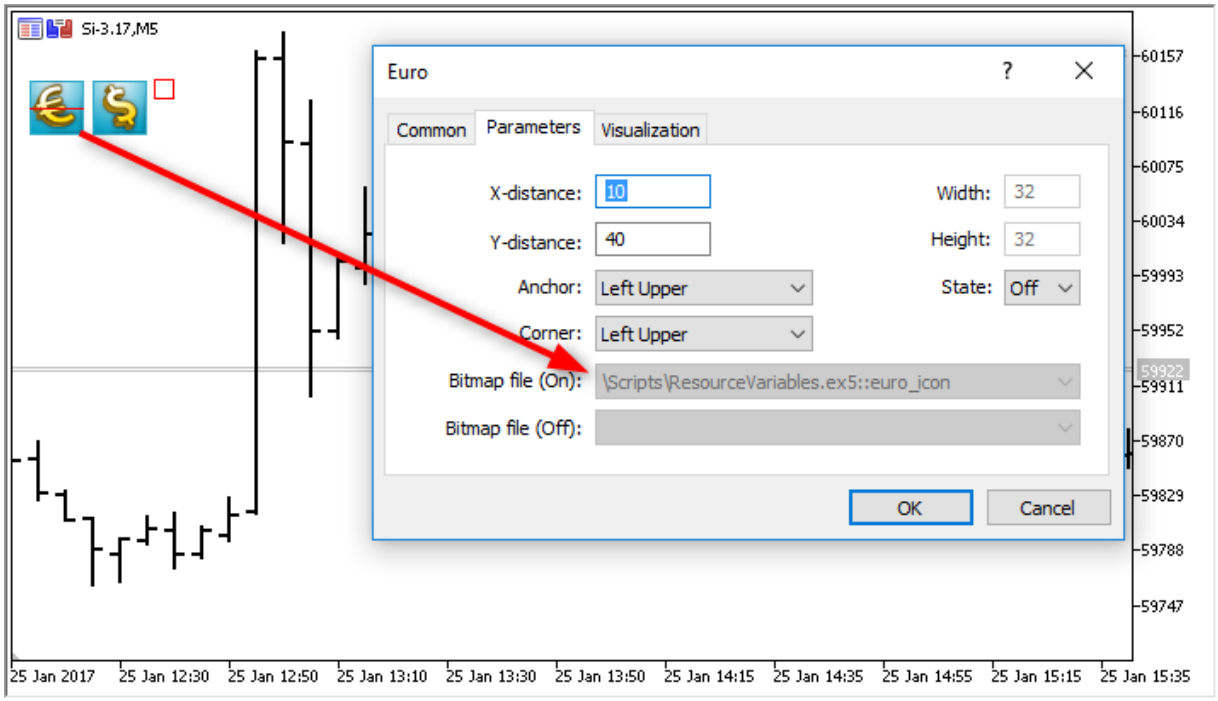
이러한 선언의 경우, 리소스 데이터는 변수를 통해서만 보내질 수 있으며 자동 어드레싱에서 "::<rsource name>"를 통해서는 작동하지 않습니다.

```
#resource "\\Images\\euro.bmp" as bitmap euro[][]
#resource "\\Images\\dollar.bmp"
//+------------------------------------------------------------------+
//|  리소스를 사용한 OBJ_BITMAP_LABEL 개체 생성 함수   |
//+------------------------------------------------------------------+
void Image(string name,string rc,int x,int y)
  {
   ObjectCreate(0,name,OBJ_BITMAP_LABEL,0,0,0);
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
   ObjectSetString(0,name,OBJPROP_BMPFILE,rc);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 유로 리소스 변수에 저장된 이미지 크기 [width, height]를 출력.
   Print(ArrayRange(euro,1),", ",ArrayRange(euro,0));
//--- 이미지를 유로로 변경 - 가운데에 빨간색 가로 줄무늬를 그림.
   for(int x=0;x<ArrayRange(euro,1);x++)
      euro[ArrayRange(euro,1)/2][x]=0xFFFF0000;
//--- 리소스 변수를 사용하여 그래픽 리소스 생성
   ResourceCreate("euro_icon",euro,ArrayRange(euro,1),ArrayRange(euro,0),0,0,ArrayRang
//--- euro_icon 리소스의 이미지를 설정할 유료 그래픽 레이블 개체를 생성
   Image("Euro","::euro_icon",10,40);
//--- 리소스를 적용하는 다른 방법으로는 그릴 수 없음
   Image("USD","::Images\\dollar.bmp",15+ArrayRange(euro,1),40);
//--- 유로 리소스 변수를 통해 이미 선언되었기 때문에 euro.bmp 리소스를 할당하는 직접 방법을 사용
   Image("E2","::Images\\euro.bmp",20+ArrayRange(euro,1)*2,40); // 실행 시간 오류가 발생
  }
```

스크립트 실행 결과 -3개의 개체 중 오직 2개의 OBJ_BITMAP_LABEL 개체가 생성됩니다. 첫 번째 개체의 이미지는 가운데에 빨간색 줄무늬가 있습니다.



리소스를 적용할 때의 중요한 이점은 리소스 파일이 컴파일 전에 실행 가능한 EX5 파일에 포함되기 전에 자동으로 압축된다는 것입니다. 따라서 리소스 변수를 사용하면 기존의 MQL5 프로그램 작성 방식에 비해 파일의 수와 전체 크기를 줄일 수 있을 뿐만 아니라 실행 가능한 EX5 파일에 필요한 모든 데이터를 직접 넣을 수 있습니다.

리소스 변수를 사용하면 마켓에서 제품을 게시하는 데 특히 편리합니다.

© 2000-2025, MetaQuotes Ltd.

Features

- 특수한 *비트맵* 리소스 변수 유형은 컴파일러에게 리소스가 이미지임을 알립니다. 그러한 변수는 uint 유형을 받습니다.
- *비트맵* 유형 배열 리소스 변수의 차원은 2개일 수 있습니다. 이 경우 어레이 크기는 [image_height][image_width]로 정의됩니다. 한 차원 배열이 지정된 경우 요소의 수는 image_height*image_width 와 같습니다.
- 24비트 이미지를 다운로드할 때 알파 채널 구성 요소는 모든 이미지 픽셀에 대해 255로 설정됩니다.
- 알파 채널이 없는 32비트 이미지를 다운로드할 때 알파 채널 구성 요소도 모든 이미지 픽셀에 대해 255로 설정됩니다.
- 알파 채널로 32비트 이미지를 다운로드할 때 픽셀은 어떤 방식으로도 처리되지 않습니다.
- 리소스 파일 크기는 128Mb를 초과할 수 없습니다.
- 문자열 파일에 대해 BOM(헤더) 존재에 의한 자동 인코딩 탐지가 수행됩니다. BOM이 없는 경우 인코딩은 파일 내용에 의해 정의됩니다. ANSI, UTF-8 및 UTF-16 인코딩의 파일이 지원됩니다. 파일에서 데이터를 읽을 때 모든 문자열이 유니코드로 변환됩니다.

OpenCL programs

리소스 문자열 변수를 사용하면 일부 프로그램의 개발을 크게 촉진할 수 있습니다. 예를 들어 별도의 CL 파일에 OpenCL 프로그램 코드를 작성한 다음 이를 문자열로 MQL5 프로그램 리소스에 포함할 수 있습니다.

```
#resource "seascape.cl" as string cl_program
...
int context;
if((cl_program=CLProgramCreate(context,cl_program)!=INVALID_HANDLE)
  {
   //--- OpenCL 프로그램을 사용하여 추가 작업 수행
  }
```

이 예제에서는 *cl_program* 리소스 변수가 사용되지 않은 경우 전체 코드를 하나의 큰 문자열로 작성해야 했습니다.

참고 항목

ResourceCreate(), ResourceSave(), PlaySound(), ObjectSetInteger(), ChartApplyTemplate(), File Functions

© 2000-2025, MetaQuotes Ltd.

임포트된 함수의 호출

mql5-프로그램 실행 중 함수를 가져오기 위해 클라이언트 터미널은 조기 바인딩을 사용합니다. 즉, 프로그램에 가져온 함수의 호출이 있는 경우 해당 모듈(ex5 또는 dll)이 프로그램 로드 중에 불러와집니다. MQL5 및 DLL 라이브러리는 호출 모듈의 스레드에서 실행됩니다.

Drive:\Directory\FileName.Ext 처럼 로드할 모듈의 전체 이름을 사용하는 것은 권장되지 않습니다. MQL5 라이브러리는 *terminal_dir\MQL5\Libraries* 폴더에서 로드됩니다. 라이브러리를 찾지 못한 경우 클라이언트 터미널에서 *terminal_dir\experts* 폴더에서 라이브러리를 로드하려고 시도합니다.

시스템 라이브러리(DLL)는 운영 체제 규칙에 의해 로드됩니다. 라이브러리가 이미 로드된 경우(예: 다른 Expert Advisor 또는 다른 클라이언트 터미널에서 병렬로 실행 중) 이미 로드된 라이브러리에 대한 요청을 사용합니다. 그렇지 않으면 다음 순서에 따라 검색을 수행합니다.

1. dll 임포트가 시작된 디렉토리입니다. 이 모듈에는 Expert Advisor, 스크립트, 지표 또는 EX5 라이브러리가 있습니다.
2. Directory terminal_data_directory\MQL5\Libraries (TERMINAL_DATA_PATH\MQL5\Libraries);
3. MetaTrader 5 클라이언트 터미널이 시작된 디렉토리;
4. 시스템 디렉토리;
5. 윈도우 디렉토리;
6. 현재 디렉토리;
7. PATH 시스템 변수에 나열된 디렉터리.

DLL 라이브러리가 작업에 다른 DLL을 사용한다면, 두 번째 DLL이 없을 경우 첫 번째 DLL을 로드할 수 없습니다.

Expert Advisor(스크립트, 인디케이터)가 로드되기 전에 모든 EX5 라이브러리 모듈의 공통 목록이 생성됩니다. 로드된 Expert Advisor(스크립트, 인디케이터)와 이 목록의 라이브러리에서 모두 사용됩니다. 따라서 여러 번 사용된 EX5 라이브러리 모듈의 1회 로드가 필요합니다. 라이브러리는 Expert Advisor(스크립트, 지표)의 미리 정의된 변수를 사용합니다.

가져온 라이브러리 EX5는 다음 순서로 검색됩니다.

1. 디렉토리, EX5를 가져오는 Expert Advisor(스크립트, 지표)의 디렉토리에 대해 설정된 경로;
2. Directory terminal_directory\MQL5\Libraries;
3. Directory MQL5₩ibraries 모든 MetaTrader 5 클라이언트 터미널의 공통 디렉토리 (Common\MQL5\Libraries).

mql5-프로그램으로 DLL을 임포트한 함수는 윈도우즈 API 호출 동의를 보장해야 합니다. 이러한 합의를 보장하기 위해, C 또는 C++로 작성된 프로그램의 소스 텍스트에서 Microsoft(r) 컴파일러 전용 키워드 __stdcall을 사용합니다. 본 동의서의 특징은 다음과 같습니다.

- 호출자(우리의 경우 mq5 프로그램)는 매개 변수를 스택에 적절하게 결합하기 위해 (DLL에서 가져온) 함수 프로토타입을 "조회"하셔야 합니다.
- 호출자(우리의 경우 mql5-프로그램)는 매개 변수를 스택에 오른쪽에서 왼쪽으로 역순으로 배치합니다. 이 순서로 가져온 함수는 전달된 매개 변수를 읽습니다.
- 매개 변수는 명시적으로 레퍼런스(우리의 경우 문자열에)에 의해 통과된 것들을 제외하고 값으로써 전달됩니다
- 가져온 함수는 스택에 전달된 매개 변수를 읽음으로써 스택을 독립적으로 정리합니다.

© 2000-2025, MetaQuotes Ltd.

가져온 함수의 프로토타입을 설명할 때 기본 파라미터를 사용할 수 있습니다.

해당 라이브러리를 로드할 수 없거나 DLL 사용에 대한 금지 사항이 있거나 가져온 기능을 찾을 수 없는 경우 – Expert Advisor는 저널(log 파일)에 "Expert Advisor stoped"라는 적절한 메시지와 함께 작업을 중지합니다. 이 경우 Expert Advisor는 다시 초기화될 때까지 실행되지 않습니다. Expert Advisor는 컴파일의 결과 또는 속성 테이블을 열고 확인을 누른 후에 다시 초기화할 수 있습니다.

Passing Parameters

단순 유형의모든 매개 변수는 명시적으로 참조로 전달되지 않는 한 값으로 전달됩니다. 문자열이 전달되면 복사된 문자열의 버퍼 주소가 전달되고, 문자열을 참조로 전달하면 복사하지 않고 이 문자열의 버퍼 주소가 DLL에서 가져온 함수에 전달됩니다.

구조–열거된 개체의 정적 또는 동적 배열 뿐만 아니라 동적 배열, 문자열, 클래스, 기타 복잡한 구조를 포함하는–를 임포트 함수에 매개 변수로 전달할 수 없습니다.

배열을 DLL에 전달할 때 데이터 버퍼 시작의 주소는 항상 전달됩니다 (AS_SERIES 플래그에 관계 없이). DLL 내부의 함수는 AS_SERIES 플래그에 대해 아무것도 알지 못합니다. 전달된 배열은 정의되지 않은 길이의 정적 배열입니다. 배열 크기를 지정하려면 추가 매개 변수를 사용해야 합니다.

© 2000–2025, MetaQuotes Ltd.

런타임 오류

클라이언트 터미널의 실행 서브시스템은 MQL5 프로그램 실행 중에 에러 코드가 발생할 경우에 대비하여 에러 코드를 저장할 수 있습니다. 각 실행 가능한 MQL5 프로그램에 대해 사전 정의된 변수 _LastError가 있습니다.

OnInit 함수를 시작하기 전에 _LastError 변수가 재설정됩니다. 계산 중 또는 내부 함수 호출 과정에서 에러 상황이 발생하는 경우 _LastError 변수는 해당 에러 코드를 받아들입니다. 이 변수에 저장된 값은 GetLastError() 함수를 사용하여 얻을 수 있습니다.

프로그램이 즉시 종료되는 경우 다음과 같은 몇 가지 심각한 오류가 발생합니다.

- 0으로 나눔
- 배열 경계를 넘어섬
- 잘못된 사용의 객체 포인터

© 2000-2025, MetaQuotes Ltd.

거래 전략 테스트

거래 로봇이 하루 24시간, 일주일 내내 쉬지 않고 작업을 할 수 있다는 점이 자동 거래의 아이디어로 어필되고 있습니다. 로봇은 지치거나 의심스럽거나 겁먹지 않습니다. 어떤 심리적 문제에서도 완전히 자유롭습니다. 거래 규칙을 명확하게 공식화해 알고리즘으로 구현하면 충분하며, 로봇은 지칠 줄 모르고 일할 준비가 돼 있습니다. 그러나 먼저 다음 두 가지 중요한 조건이 충족되는지 확인해야 합니다:

- Expert Advisor는 거래 시스템의 규칙에 따라 거래 작업을 수행합니다;
- EA에서 시행된 거래 전략은 그간의 내역에서 이익을 있음을 보여줍니다.

이러한 질문에 대한 답변을 얻으려면 MetaTrader 5 클라이언트 터미널에 포함된 Strategy Tester로 이동합니다.

이 섹션에서는 전략 테스터의 프로그램 테스트 및 최적화 기능에 대해 설명합니다:

- 전략 테스터의 함수 제한
- 틱 생성 모드
- 확산 시뮬레이션
- 테스트 중 실제 틱 사용
- 클라이언트 터미널의 글로벌 변수
- 테스트 중 지표 계산
- 테스트 중 내역 로드
- 다중 통화 테스트
- 전략 테스터의 시간 시뮬레이션
- 테스트의 그래픽 개체
- 전략 테스터의 OnTimer() 함수
- 전략 테스터의 Sleep() 함수
- 수학 계산에서 최적화 문제를 위한 전략 테스터 사용
- "개방 가격 전용" 모드의 막대 동기화
- 테스터의 IndicatorRelease() 함수
- 테스터의 이벤트 핸들링
- 테스팅 에이전트
- 터미널과 에이전트 간의 데이터 교환
- 모든 클라이언트 터미널의 공유 폴더 사용
- DLLs 사용

MQL5 Cloud Network의 메모리 및 디스크 공간 제한

다음 제한 사항은 MQL5 Cloud Network 최적화 작업 실행시에 적용됩니다.: Expert Advisor는 4GB 이상의 정보를 디스크에 쓰거나 4GB 이상의 RAM을 사용해서는 안 됩니다. 한도를 초과하면 네트워크 에이전트가 계산을 올바르게 완료할 수 없으며 열러분은 결과를 받을 수 없습니다. 그러나 계산에 소요된 모든 시간에 대해 비용이 청구됩니다.

© 2000-2025, MetaQuotes Ltd.

각각의 최적화 단계에서 정보를 가져와야 하는 경우 디스크에 쓰지 말고 프레임을 보내세요 . MQL5 Cloud Network에서 계산하는 동안 Expert Advisors에서 파일 작업을 피하려면 다음과 같은 확인을 사용할 수 있습니다.

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

전략 테스터의 함수 제한

클라이언트 터미널의 Strategy Tester에는 일부 함수에 대한 작동 제한이 있습니다.

Comment(), Print() 및 PrintFormat() 함수

퍼포먼스 향상을 위한 Comment(), Print() 및 PrintFormat()함수는 거래 로봇의 매개 변수를 최적화할 때 실행되지 않습니다. OnInit() 핸들러 내부에서 이러한 함수를 사용하는 것은 예외입니다. 이렇게 하면 오류가 발생할 때 쉽게 오류를 찾을 수 있습니다.

Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest() 함수

"외부 세계"와의 상호작용을 위해 설계된 Alert(), MessageBox(), PlaySound(), SendFTP(), SendMail(), SendNotification() 및 WebRequest() 함수는 략 테스터에서 실행되지 않습니다.

틱 생성 모드

Expert Advisor는 MQL5로 작성된 프로그램으로, 외부 이벤트에 대응하여 매번 실행됩니다. EA에는 사전 정의된 각 이벤트에 해당하는 함수(이벤트 핸들러)가 있습니다.

NewTick 이벤트(가격 변동)는 EA의 주요 이벤트이므로 EA를 테스트하기 위한 틱 시퀀스를 생성해야 합니다. MetaTrader 5 클라이언트 터미널의 Strategy Tester에는 다음과 같은 세 가지 틱 생성 모드가 구현되어 있습니다:

- Every tick
- 1분 OHLC(분막대와 OHLC 가격)
- 시가 전용

© 2000-2025, MetaQuotes Ltd.

기본이자 가장 상세한 모드는 "Every tick" 모드이며, 나머지 두 모드는 "Every tick" 모드와 비교하여 설명됩니다. 세 가지 모드 간의 차이를 이해하기 위해 모두를 고려하십시오.

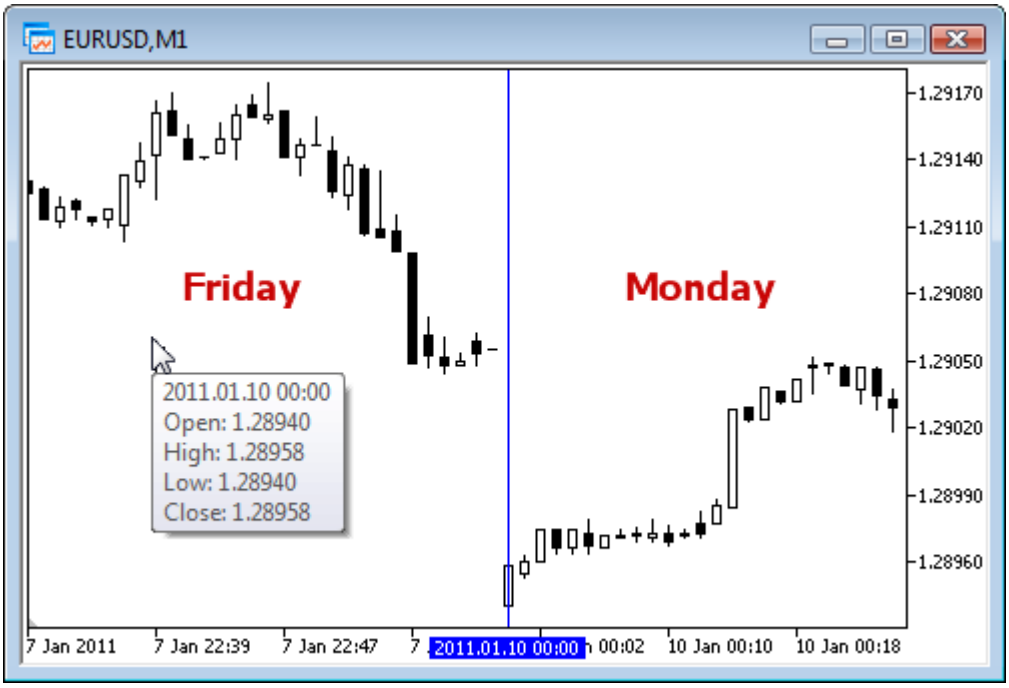
"Every Tick"

금융상품의 과거 시세 데이터는 거래 서버에서 MetaTrader 5 클라이언트 터미널로 분량 막대 형태로 전송됩니다. 요청의 발생과 필요한 타임프레임 구성에 대한 자세한 정보는 MQL5 참조의 데이터 액세스 구성 장에서 얻을 수 있습니다.

가격 내역의 최소 요소는 분 막대이며, 여기서 다음 네 가지 가격 값에 대한 정보를 얻을 수 있습니다.

- 시가 – 분 단위 막대가 오픈된 가격;
- 고가 – 이 분 막대에서 달성한 최대치;
- 저가 – 이 분 막대에서 달성한 최소값;
- 종가 – 바의 마감 가격.

새로운 분 막대는 새 분기가 시작되는 순간에 오픈되는 것이 아니라(초 수가 00이 됨), 틱이 발생할 때 – 가격이 최소 1포인트 이상 변경됩니다. 이 수치는 2011.01.10 00:00의 개장 시간을 가진 새로운 거래 주의 첫 번째 분 막대를 보여줍니다. 우리가 차트에서 보는 금요일과 월요일의 가격 차이는 흔히 볼 수 있는데, 이는 들어오는 뉴스에 반응하여 주말에도 환율이 변동하기 때문입니다.



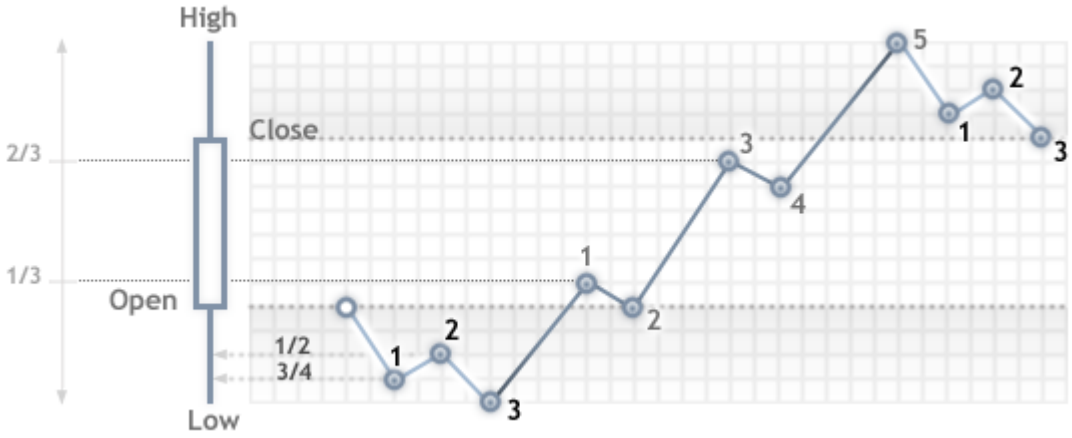
이 바의 경우 2011년 1월 10일 00시간 00분에 분 막대가 오픈되었다는 것만 알 수 있을 뿐 초는 알 수 없습니다. 00:00:12 또는 00:00:36(다음 날 시작 후 12초 또는 36초) 또는 해당 분 내에 다른 시간에 오픈되었을 수 있습니다. 그러나 우리는 EURUSD의 오픈 가격이 새 분 막대의 오프닝 시간에 1.28940에 있었다는 것을 알 수 있습니다.

또한 고려된 분 막대의 종가에 해당하는 틱을 언제 받았는지 알 수 없습니다. 또한 고려된 분 막대의 종가에 해당하는 틱을 언제 받았는지 알 수 없습니다. 이 분 동안, 가격은 1.28958이었습니다. 고가와 저가의 등장 시점도 알 수 없지만, 최고가와 최저가가 각각 1.28958, 1.28940 수준에 그친 것으로 알 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

거래 전략을 테스트하기 위해서는 Expert Advisor의 작업을 시뮬레이션할 일련의 틱이 필요합니다. 따라서, 매 분마다 가격이 확실히 존재했던 **4개의 제어 지점**을 알 수 있습니다. 막대에 4개의 틱만 있으면 테스트를 수행하기에 충분한 정보이고, 일반적으로 눈금 볼륨이 4개보다 큽니다.

따라서 시가, 고가, 저가, 및 종가 사이에서 발생한 눈금에 대한 추가 관리 지점을 생성해야 합니다. "Every tick" 생성 모드의 원리는 MetaTrader 5 Terminal의 Strategy Tester에 있는 Ticks 생성 알고리즘에 설명되어 있으며, 아래에 나와 있습니다.



"Every tick" 모드에서 테스트할 때 EA의 OnTick() 함수가 모든 제어 지점에서 호출됩니다. 각 제어점은 생성된 시퀀스의 틱입니다. EA는 온라인 작업 시처럼 시뮬레이션 틱의 시간과 가격을 받게 됩니다.

> **중요:** "Every tick" 테스트 모드가 가장 정확하면서도 동시에 시간이 가장 많이 소요됩니다. 대부분의 거래 전략의 초기 시험의 경우, 보통 나머지 두 가지 시험 모드 중 하나를 사용하는 것으로 충분합니다.

"1 분 OHLC"

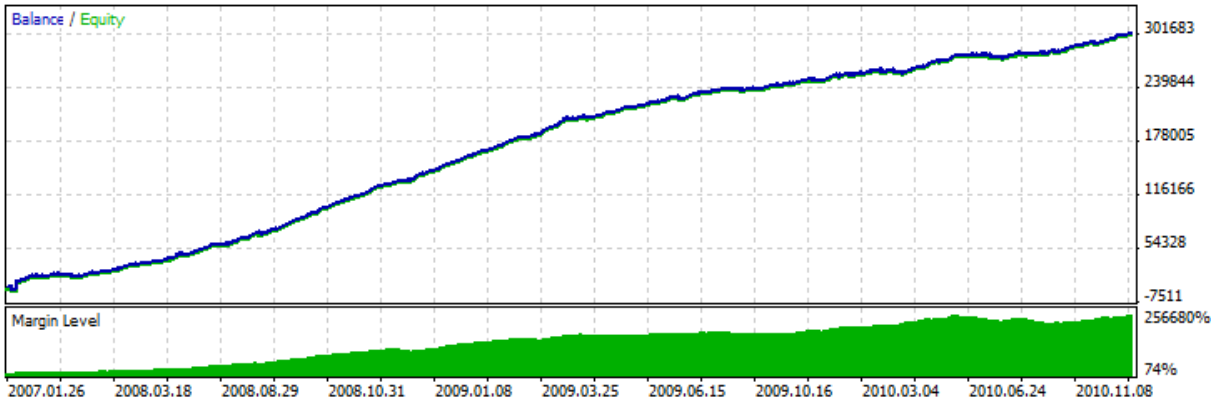
"Every tick" 모드는 세 가지 모드 중 가장 정확하지만 동시에 가장 느립니다. OnTick() 핸들러는 모든 틱에서 실행되지만 틱 볼륨은 상당히 클 수 있습니다. 막대 전체에 걸친 가격 이동의 틱 순서가 중요하지 않은 전략의 경우, 더 빠르고 대략적인 시뮬레이션 모드인 "1분 OHLC"가 있습니다.

"1분 OHLC" 모드에서 틱 시퀀스는 **분봉의 OHLC 가격**에 의해서만 생성되며, 생성된 제어 지점의 수는 상당히 감소하므로 테스트 시간도 감소합니다. OnTick() 기능의 시작은 OHLC minute bar의 가격에 의해 구성된 모든 제어 지점에서 수행됩니다.

시가, 고가, 저가, 종가 사이에 추가적인 중간 틱이 생성되지 않는 것은 시가가 결정되는 순간부터 가격 개발에 있어 엄격한 결정론의 출현으로 이어집니다. 이렇게 하면 "Testing Grail"을 만들 수 있으며, 이 그래프는 테스트 밸런스의 좋은 위쪽 그래프를 보여줍니다.

CodeBase에는 이러한 Grail의 예가 나와 있습니다 -Grr-al.

© 2000-2025, MetaQuotes Ltd.



그림은 이 EA 테스트의 매우 매력적인 그래프를 보여줍니다. 어떻게 얻었을까요? 우리는 1분 막대의 4가지 가격을 알고 있으며, 첫 번째는 시가이고 마지막은 종가라는 것도 알고 있습니다. 그것들 사이에 고가와 저가가 있으며, 발생 순서는 알 수 없지만, 고가가 시가보다 높거나 같은 것으로 알려져 있습니다.

공개 가격을 받는 순간을 결정한 뒤 다음 틱 표시를 분석하여 현재 보유하는 가격(높은 가격 또는 낮은 가격)을 결정하기에 충분합니다. 만약 가격이 시가보다 낮고, 저가로 이 틱에서 구매하면, 다음 틱은 고가에 해당할 것이며, 저희가 여기서 구매를 끝내고 판매를 위해 오픈할 것입니다. 다음 틱이 마지막이고, 이것이 종가이며, 판매는 끝납니다.

그 가격 이후에 시가 가격보다 큰 금액의 틱이 나오면 거래 순서가 뒤바뀝니다. 이 "치트" 모드에서 1분 막대를 처리하고 다음 막대를 기다립니다.

그러한 EA를 역사에 대해 시험할 때, 모든 것은 순조롭게 진행되지만, 일단 저희가 그것을 온라인으로 시작하면, '균형선은 흔들리지 않고, 다만 아래로 향한다'는 진실이 밝혀지기 시작합니다. 이 트릭을 노출시키려면 "Every tick" 모드로 EA를 실행하면 됩니다.

> **참고:** 대략적인 테스트 모드("1분 OHLC" 및 "개방 가격 전용")에서 EA의 테스트 결과가 너무 좋은 것 같으면 "Every tick" 모드에서 테스트해야 합니다.

"시가 전용"

이 모드에서는 테스트를 위해 선택한 기간의 OHLC 가격에 따라 틱이 생성됩니다. Expert Advisor의 OnTick() 함수는 바 시작 시 시가에서만 실행됩니다. 이 기능으로 인해 중지 레벨 및 보류는 지정된 가격과 다른 가격으로 트리거될 수 있습니다(특히 더 높은 타임프레임에서 테스트하는 경우). 대신 Expert Advisor의 평가 테스트를 신속하게 실행할 수 있습니다.

W1 및 MN1 기간은 "시가 전용" 틱 생성 모드의 예외입니다. 이러한 기간 동안 타임프레임 틱은 해당 주 또는 달의 OHLC 가격이 아니라 각 요일의 OHLC 가격에 대해 생성됩니다.

EURUSD H1에서 "시가 전용" 모드에서 Expert Advisor를 테스트한다고 가정합니다. 이 경우 총 틱(제어 지점) 수는 검정된 간격 내에 있는 한 시간 막대의 4*개보다 작지 않습니다. 하지만 **OnTick() 핸들러는 시봉(1시간 막대)의 시작 시에만 호출됩니다.**. 올바른 테스트에 필요한 검사는 나머지 틱(EA로부터 "숨겨진") 에서 수행됩니다.

- 마진 요구사항 계산;
- 스탑 로스 및 이익 실현 촉발 수준;
- 보류 주문의 트리거;
- 만료된 보류 주문 제거.

© 2000-2025, MetaQuotes Ltd.

오픈 포지션나 보류 주문이 없는 경우 숨겨진 틱에 대해 이러한 검사를 수행할 필요가 없으며 속도가 상당히 빨라질 수 있습니다. 이 "시가 전용" 모드는 바 개설 시에만 거래를 처리하고 보류 주문은 사용하지 않는 테스트 전략과 스탑 로스 및 이익 실현 주문에 적합합니다. 이러한 전략의 등급에 필요한 테스트 정확도는 보존됩니다.

모든 모드에서 테스트할 수 있는 EA의 예로 표준 패키지의 이동 평균 Expert Advisor를 사용하겠습니다. 이 EA의 논리는 모든 결정이 막대의 개시에서 이루어지며, 보류 주문을 사용하지 않고 즉시 거래가 진행될 수 있도록 구축됩니다.

2010.09.01에서 2010.12.31까지의 기간에 EURUSD H1에 대한 EA 테스트를 실행하고 그래프를 비교합니다. 그림은 세 가지 모드 모두에 대한 테스트 보고서의 균형 그래프를 보여줍니다.

보시다시피, 다른 테스트 모드의 그래프는 표준 패키지의 이동 평균 EA와 동일합니다.

"시가 전용" 모드에는 다음과 같은 몇 가지 제한이 있습니다.

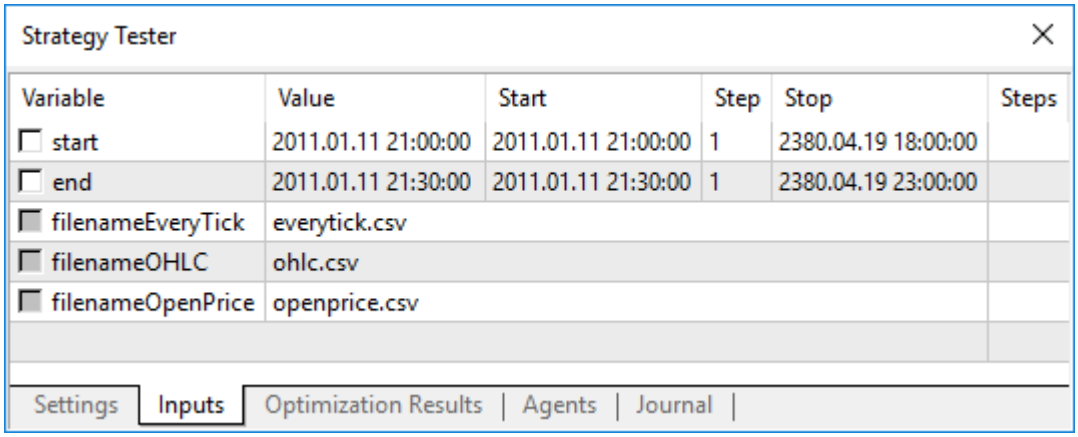
- 다음을 사용할 수 없습니다: 임의 지연 실행 모드.
- 테스트된 Expert Advisor에서는 테스트/최적화에 사용된 시간보다 낮은 타임프레임의 데이터에 액세스할 수 없습니다. 예를 들어, H1 기간에 테스트/최적화를 실행하면 H2, H3, H4 등의 데이터에 액세스할 수 있지만 M30, M20, M10 등의 데이터는 액세스할 수 없습니다. 또한 액세스되는 더 높은 타임프레임은 테스트 시간의 몇 배여야 합니다. 예를 들어 M20에서 테스트를 실행하면 M30의 데이터에 액세스할 수 없지만 H1에 액세스할 수 있습니다. 이러한 제한은 테스트/최적화 중에 생성된 막대에서 낮거나 다중이 아닌(non-multiple) 타임프레임의 데이터를 얻을 수 없는 것과 관련이 있습니다.
- 다른 타임프레임의 데이터 액세스에 대한 제한은 Expert Advisor가 데이터를 사용하는 다른 기호에도 적용됩니다. 이 경우 각 심볼에 대한 제한은 테스트/최적화 중에 액세스하는 첫 번째 시간에 따라 달라집니다. EURUSD H1에서 테스트하는 동안 Expert Advisor가 GBPUSD M20의 데이터에 액세스한다고 가정해보십시오. 이 경우 Expert Advisor는 EURUSD H1, H2 등의 데이터와 GBPUSD M20, H1, H2 등의 데이터를 추가로 사용할 수 있습니다.

> **참고:** "시가 전용" 모드는 테스트 시간이 가장 빠르지만 모든 거래 전략에 적합하지 않습니다. 거래 시스템의 특성에 따라 원하는 테스트 모드를 선택하십시오.

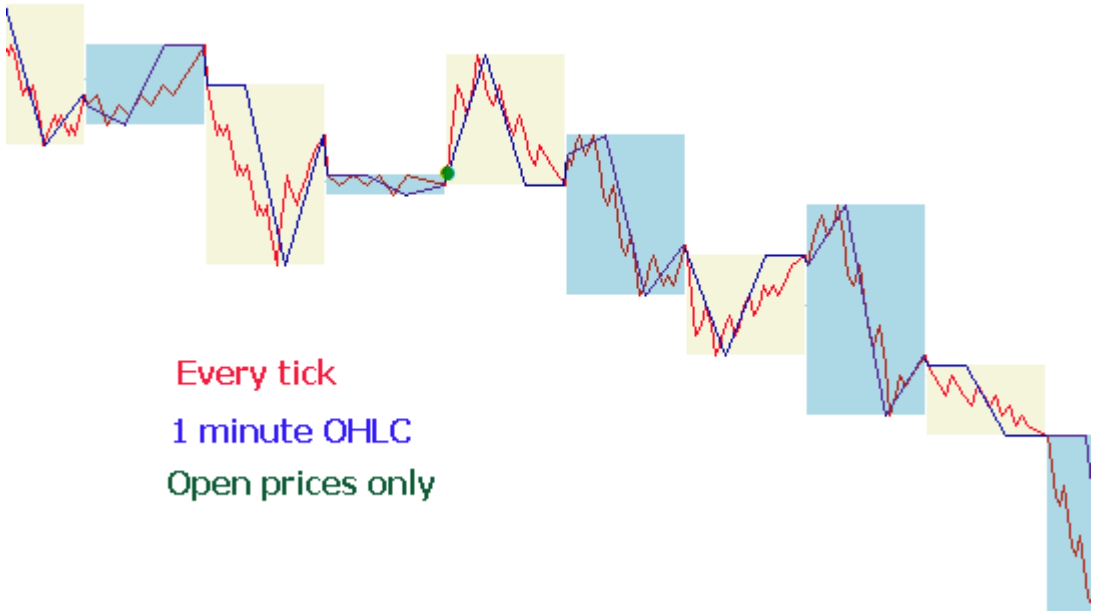
© 2000-2025, MetaQuotes Ltd.

틱 생성 모드에 대한 섹션을 마무리하려면 2011.01.11 21:00:00 – 2011.01.11 21:30:00 간격의 두 M15 막대에 대한 EURUSD에 대한 여러 틱 생성 모드의 시각적 비교를 고려해보십시오.

틱들은 WriteTicksFromTester.mq5 EA를 사용하여 다른 파일에 저장되었으며 파일 이름의 끝은 filenameEveryTick, filenameOHLC 및 filenameOpenPrice 입력 매개 변수에 지정되어 있습니다.



3개의 틱 시퀀스가 있는 파일 3개를 얻기 위해(다음 모드 각각에 대해 "Every tick", "1분 OHLC", "시가 전용") EA는 해당 모드에서 단일 실행으로 3번 실행되었습니다. 그런 다음 이 세 파일의 데이터가 TicksFromTester.mq5 지표를 사용하여 차트에 표시되었습니다. 이 물품에는 지표 코드가 첨부되어 있습니다.



기본적으로, MQL5 언어의 모든 파일 작업은 "파일 샌드박스" 내에서 이루어지며, 테스트 중에 EA는 자신의 "파일 샌드박스"에만 액세스할 수 있습니다. 테스트 중에 지표와 EA가 한 폴더의 파일로 작동하기 위해 flag FILE_COMMON을 사용했습니다. EA의 코드 예제:

```
//--- 파일 오픈
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- 파일 핸들 점검
   if(file==INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("%s 파일 작성을 위해 여는 중 오류 발생. Error code=%d",filename,GetLastE
       return;
      }
    else
      {
       PrintFormat("파일이 %s 폴더에 생성됩니다",TerminalInfoString(TERMINAL_COMMONDATA_PA
      }
```

지표의 데이터를 읽기 위해 FILE_COMMON 플래그도 사용했습니다. 이를 통해 필요한 파일을 폴더 간에 수동으로 전송하는 것을 방지할 수 있었습니다.

```
//--- 파일 오픈
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- 파일 핸들 점검
   if(file==INVALID_HANDLE)
      {
       PrintFormat("%s 파일을 읽기 위해 오픈 중 오류 발생. Error code=%d",fname,GetLastErro
       return;
      }
   else
      {
       PrintFormat("파일이 %s 에서 열립니다",TerminalInfoString(TERMINAL_COMMONDATA_PATH)
      }
```

확산 시뮬레이션

입찰 가격과 요청 가격 간의 가격 차이를 스프레드라고 합니다. 시험 중 스프레드는 모델링되지 않지만 과거 데이터에서 추출됩니다. 과거 데이터에서 산포도가 0보다 작거나 같으면 마지막으로 알려진(생성 순간) 산포도가 테스트 에이전트에서 사용됩니다.

Strategy Tester에서 스프레드는 항상 floating으로 간주됩니다. 즉, SymbolInfoInteger(기호, SYMBOL_SPREAD_FLOAT)는 항상 true를 반환합니다.

또한 내역 데이터에는 틱 값과 거래량이 포함됩니다. 데이터의 저장 및 검색을 위해 특별한 MqlRates 구조를 사용합니다:

```
struct MqlRates
  {
   datetime time;         // 기간 시작 시간
   double   open;         // 시가
   double   high;         // 그 기간의 최고 가격
   double   low;          // 해당 기간의 최저 가격
   double   close;        // 종가
   long     tick_volume;  // 틱 볼륨
   int      spread;       // 스프레드
   long     real_volume;  // 거래량
  };
```

테스트 중 실제 틱 사용

© 2000-2025, MetaQuotes Ltd.

실제 틱에 대한 테스트 및 최적화는 가능한 실제 조건에 가깝습니다. 분 데이터를 기반으로 생성된 틱 대신 브로커가 누적한 실제 틱을 사용할 수 있습니다. 이것들은 거래소와 유동성 제공자들의 틱입니다.

최대 테스트 정확도를 보장하기 위해 실제 틱 모드에서도 분봉이 사용됩니다. 막대는 틱 데이터를 확인하고 수정하는 데 적용됩니다. 또한 테스터와 클라이언트 터미널에서 차트가 분산되는 것을 방지할 수 있습니다.

테스터는 틱 데이터를 분봉 매개변수와 비교합니다. 틱이 막대의 높음/낮음 수준을 초과하면 안 되며, 초기 및 최종 틱이 막대의 시가/종가와 일치해야 합니다. 볼륨도 비교됩니다. 불일치가 탐지되면 이 분봉에 해당하는 모든 틱이 삭제됩니다. 생성된 틱은 대신 사용됩니다(예: "모든 틱" 모드).

심볼 내역에 틱 데이터가 없는 분봉이 있는 경우 테스터는 "Every tick" 모드에서 눈금을 생성합니다. 이렇게 하면 브로커의 틱 데이터가 충분하지 않은 경우 테스터에서 올바른 차트를 플로팅할 수 있습니다.

심볼 내역에 분봉이 없지만 분에 해당하는 틱 데이터가 있는 경우 테스터에서 데이터를 사용할 수 있습니다. 예를 들어, 거래 심볼 페어는 종가를 사용하여 형성됩니다. 종가가 서버에서 도착하지 않은 상태에서 입찰/요청 가격에 대한 틱만 있다면 막대가 생성되지 않습니다. 테스터는 이러한 틱 데이터가 분봉과 모순되지 않으므로 이 틱 데이터를 사용합니다.

소스에서 클라이언트 터미널로 데이터를 전송할 때 연결 끊김 또는 기타 장애로 인해 틱 데이터가 여러 가지 이유로 인해 분봉과 일치하지 않을 수 있습니다. 분당 데이터는 테스트 중에 더 신뢰할 수 있는 것으로 간주됩니다.

실제 틱을 테스트할 때 다음 기능에 유의하십시오:

- 테스트를 시작할 때 심볼에 있는 분 데이터가 틱 데이터와 동기화됩니다.
- 틱들은 전략 테스터의 심볼 캐시에 저장됩니다. 캐시 크기가 128000 틱을 초과하지 않습니다. 새 틱이 도착하면 가장 오래된 데이터가 캐시에서 제거됩니다. 그러나, CopyTicks 함수는 캐시 외부에서 틱 수신을 허용합니다(실제 틱에 대해 테스트하는 경우에만). 이 경우 해당 클라이언트 터미널 데이터베이스와 완전히 유사한 테스터 틱 데이터베이스에서 데이터를 요청합니다. 이 베이스에는 분 막대 보정이 구현되지 않습니다. ㅁ따라서, 해당 틱들은 캐시에 저장된 틱들과 다를 수 있습니다.

클라이언트 터미널의 글로벌 변수

테스트 과정에서 클라이언트 터미널의 글로벌 변수'도 에뮬레이션되지만, F3 버튼을 이용해 단말에서 볼 수 있는 현재의 터미널의 글로벌 변수와는 관련이 없습니다. 즉, 테스트 중에 터미널의 전역 변수를 사용하는 모든 작업이 클라이언트 터미널(테스트 에이전트) 외부에서 수행됩니다.

테스트 중 지표 계산

실시간 모드에서 지표값은 매 틱마다 계산됩니다.

전략 테스터에서 지표가 계산되는 방식은 데이터 생산을 위해 액세스되는 경우, 즉 지표의 버퍼 값이 요청되는 경우에만 계산됩니다. 유일한 예외는 커스텀 지표가 #property tester_everytick_calculate로 지정된 경우 뿐입니다. 이 경우 각 틱에 대해 재계산이 수행됩니다.

시각적 테스트 모드에서는 모든 지표는 시각적 테스트 차트에 올바르게 표시되기 위해 새 틱이 도착할 때 무조건 다시 계산됩니다.

지표는 틱당 한 번 계산됩니다. 지표의 데이터에 대한 모든 후속적인 요청은 새 틱이 도착할 때까지 재계산하지 않습니다. 따라서 EA에서 EventSetTimer() 함수를 통해 타이머가 활성화된 경우 지표의 데이터는 OnTimer() 핸들러의 각 호출 이전의 마지막 틱부터 요청됩니다. 지표가 마지막 틱에서 아직 계산되지

© 2000-2025, MetaQuotes Ltd.

않은 경우 지표 값 계산이 시작됩니다. 데이터가 이미 준비된 경우에는 새롭게 재계산 하지 않지 않고 제공됩니다.

따라서 모든 지표의 계산은 가장 리소스를 절약하는 방식으로 수행됩니다. 지표가 지정된 틱에서 이미 계산된 경우에는 해당 데이터는 '있는 그대로' 제공됩니다. 재계산을 하지 않습니다.

테스트 중 내역 로드

테스트 프로세스를 시작하기 전에 거래 서버에서 터미널에 의해 테스트될 심볼의 내역이 동기화되고 로드됩니다. 첫 번째 시간 동안 터미널은 나중에 요청하지 않기 위해 사용 가능한 모든 심볼 내역을 로드합니다. 또한 새 데이터만 로드됩니다.

테스트 에이전트는 테스트 시작 직후 클라이언트 터미널에서 테스트할 기호 심볼을 수신합니다. 다른 금융상품의 데이터가 테스트 과정에서 사용되는 경우(예: 다중 통화 Expert Advisor), 테스트 에이전트는 이러한 데이터를 처음 호출하는 동안 클라이언트 터미널에서 필요한 내역을 요청합니다. 터미널에서 사용 가능한 내역 데이터가 있으면 즉시 테스트 에이전트로 전달됩니다. 데이터를 사용할 수 없는 경우 터미널에서 요청하여 서버에서 다운로드한 다음 테스트 에이전트로 전달합니다.

무역업무에 대한 교차요율을 산정하기 위해서는 추가 금융상품의 데이터도 필요합니다. 예를 들어, 첫 번째 거래 작업을 처리하기 전에, USD의 예금 통화로 EURUCHF에 대한 전략을 테스트할 때, 테스트 에이전트는 클라이언트 터미널에서 EURUSD와 USDCHF의 이력 데이터를 요청하지만, 이러한 심볼들의 직접 사용 호출은 포함하지 않습니다.

다중 통화 전략을 테스트하기 전에 필요한 모든 과거 데이터를 클라이언트 터미널로 다운로드하는 것이 좋습니다. 이렇게 하면 필요한 데이터 다운로드와 관련된 테스트/최적화가 지연되는 것을 방지할 수 있습니다. 예를 들어, 적절한 차트를 열고 내역 시작 부분으로 스크롤하여 기록을 다운로드할 수 있습니다. MQL5 레퍼런스의 데이터에 대한 액세스 구성 섹션에서 내역을 터미널에 강제로 로드하는 예를 볼 수 있습니다.

테스트 에이전트는 터미널에서 패킹된 형태로 내역을 수신합니다. 다음 테스트 중에는 테스터가 이전에 테스터를 실행한 이후 필요한 데이터를 사용할 수 있기 때문에 테스터가 터미널에서 이력을 로드하지 않습니다.

- 터미널은 에이전트가 터미널에서 테스트 할 심볼 내역을 처음 요청할 때 한 번만 트레이드 서버에서 기록을 로드합니다. 내역은 트래픽을 줄이기 위해 꽉 찬 형태로 적재됩니다.
- 틱은 네트워크를 통해 전송되지 않고 테스트 에이전트에서 생성됩니다.

다중 통화 테스트

Strategy Tester를 사용하면 여러 심볼을 사용하여 전략을 테스트할 수 있습니다. 원래 이전 플랫폼에서는 단일 심볼에 대해서만 테스트가 수행되었기 때문에 이러한 EA는 전통적으로 다중 통화 Expert Advisor로 불립니다. MetaTrader 5 터미널의 Strategy Tester에서는 사용 가능한 모든 심볼에 대한 거래를 모델링할 수 있습니다.

테스터는 기호 데이터를 처음 호출하는 동안 **클라이언트 터미널**(거래 서버가 아님!)에서 사용된 기호 내역을 자동으로 로드합니다.

테스트 에이전트는 테스트 시작 시 지표 계산에 필요한 내역 데이터를 제공하기 위해 적은 여백으로 누락된 기록만 다운로드합니다. D1 이하의 기간 동안 다운로드된 내역의 최소 볼륨은 1년입니다.

© 2000-2025, MetaQuotes Ltd.

따라서 M15 기간(각 막대는 15분)으로 2010.11.01-2010.12.01(1개월 간격 테스트)에 대한 테스트를 실행하면, 2010년 전체 금융상품 이력이 요청됩니다. 주간 타임프레임의 경우 약 2년(연간 52주)에 해당하는 100개의 막대의 이력을 요청할 예정입니다. 월별 타임프레임에 대한 테스트의 경우 에이전트는 8년(12개월 x 8년 = 96개월)의 내역을 요청합니다.

필요한 막대가 없는 경우 **테스트 시작 날짜가 자동으로** 과거에서 현재로 전환되어 테스트 전에 필요한 막대 예약 기능을 제공합니다.

테스트 중에 "Market Watch"도 에뮬레이션 되어 심볼에 대한 정보를 얻을 수 있습니다.

기본적으로 테스트를 시작할 때 전략 테스터의 "Market Watch"에는 테스트가 실행 중인 심볼인 단 하나의 심볼이 있습니다. 필요한 모든 심볼은 전략 테스터의 "Market Watch"(터미널이 아님)에 자동으로 연결됩니다.

> 다중 통화 Expert Advisor의 테스트를 시작하기 전에 터미널의 "Market Watch"에서 테스트에 필요한 심볼을 선택하고 필요한 데이터를 로드해야 합니다. "해외" 심볼의 첫 번째 호출 동안, 그것의 내역은 테스트 에이전트와 클라이언트 터미널 간에 자동으로 동기화될 것입니다. "외부" 심볼은 테스트가 실행 중인 심볼 이외의 심볼입니다.

"외부" 심볼에 대한 데이터 참조는 다음과 같은 경우에 발생합니다:

- 심볼/타임프레임에 기술적 지표 함수 및 IndicatorCreate()를 사용하는 경우;
- 다른 심볼에 대한 "Market Watch" 데이터에 대한 요청:
  1. SeriesInfoInteger
  2. Bars
  3. SymbolSelect
  4. SymbolIsSynchronized
  5. SymbolInfoDouble
  6. SymbolInfoInteger
  7. SymbolInfoString
  8. SymbolInfoTick
  9. SymbolInfoSessionQuote
  10. SymbolInfoSessionTrade
  11. MarketBookAdd
  12. MarketBookGet
- 다음 함수를 사용하여 심볼/타임프레임에 대한 시계열 요청:
  1. CopyBuffer
  2. CopyRates
  3. CopyTime
  4. CopyOpen
  5. CopyHigh
  6. CopyLow
  7. CopyClose
  8. CopyTickVolume

© 2000-2025, MetaQuotes Ltd.

9. CopyRealVolume
10. CopySpread

"다른" 심볼로 처음 호출하는 순간, 테스트 프로세스가 중지되고 터미널에서 테스트 에이전트로 심볼/타임프레임에 대한 내역이 다운로드됩니다. 동시에 이 심볼에 대한 틱 시퀀스가 생성됩니다.

선택한 틱 생성 모드에 따라 각 심볼에 대해 개별 틱 시퀀스가 생성됩니다. OnInit() 핸들러에서 SymbolSelect()를 호출하여 원하는 심볼을 위해 내역을 명시적으로 요청할 수도 있습니다 – 기록 다운로드는 Expert Advisor 테스트 직전에 이루어집니다.

따라서 MetaTrader 5 클라이언트 터미널에서 다중 통화 테스트를 수행하는 데 추가적인 노력이 필요하지 않습니다. 클라이언트 터미널에서 해당 심볼 차트를 열기만 하면 됩니다. 이 데이터가 들어 있는 경우 필요한 모든 심볼에 대한 내역이 거래 서버에서 자동으로 업로드됩니다.

전략 테스터의 시간 시뮬레이션

테스트 중 현지시간 TimeLocal()는 항상 서버시간 TimeTradeServer()과 같습니다. 차례로, 서버 시간은 항상 GMT 시간 – TimeGMT()에 해당하는 시간과 같습니다. 이렇게 하면 이러한 모든 함수가 테스트 중에 동일한 시간을 표시합니다.

서버에 연결되지 않은 경우 전략 테스터에 GMT, 로컬 및 서버 시간 차이가 없기 때문에 신중히 수행됩니다. 연결 여부에 관계없이 테스트 결과는 항상 동일해야 합니다. 서버 시간에 대한 정보는 로컬에 저장되지 않으며 서버에서 가져옵니다.

테스트의 그래픽 개체

테스트/최적화 중에는 그래픽 개체가 표시되지 않습니다. 따라서 테스트/최적화 중 생성된 객체의 속성을 참조할 때 Expert Advisor는 0 값을 받게 됩니다.

> 비쥬얼 모드에서 테스트하는 경우에는 이 제한이 적용되지 않습니다.

전략 테스터의 OnTimer() 함수

MQL5는 타이머 이벤트를 처리할 수 있는 기회를 제공합니다. 테스트 모드에 관계없이 OnTimer() 핸들러의 호출이 수행됩니다. 즉, 테스트가 H4 기간 동안 "시가 전용" 모드에서 실행 중이고 EA가 초당 통화로 설정된 타이머를 가진 경우 각 H4 막대가 열릴 때 OnTick() 핸들러를 한 번 호출하고 OnTimer() 핸들러를 14400회(3600초 * 4시간) 호출합니다. EA의 테스트 시간이 증가하는 양은 EA의 논리에 따라 달라집니다.

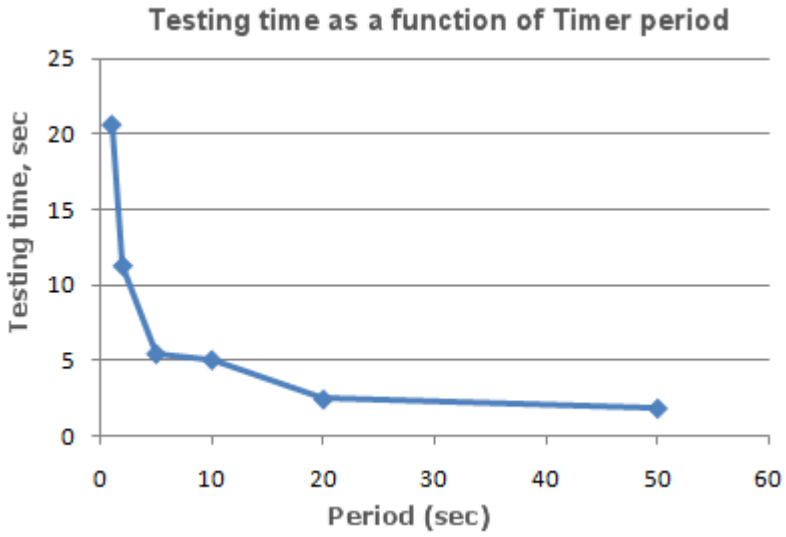
타이머의 주파수에 따른 테스트 시간의 의존성을 확인하기 위해 거래 작업 없이 간단한 EA를 만들었습니다.

```
//--- 입력 매개 변수
input int      timer=1;              // 타이머 값, sec
input bool     timer_switch_on=true; // 타이머 온
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- run the timer if  timer_switch_on==true
```

© 2000-2025, MetaQuotes Ltd.

```
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 취소 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 타이머 정지
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| 타이머 함수                                                      |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// 핸들러 본문이 비어 있으므로 아무 조치도 취하지 않습니다
  }
//+------------------------------------------------------------------+
```

테스트 시간 측정은 타이머 파라미터의 다른 값(Timer 이벤트의 주기)에서 수행되었습니다. 얻은 데이터에 대해 타이머 기간의 함수로써 테스트 시간을 플로팅합니다.



동일한 다른 조건에서 EventSetTimer(타이머) 함수를 초기화하는 동안, 타이머 매개 변수가 작을수록, OnTimer() 핸들러의 호출 간 주기(주기)가 작을수록, 테스트 시간 T가 더 크다는 것을 명확히 알 수 있습니다.

전략 테스터의 Sleep() 함수

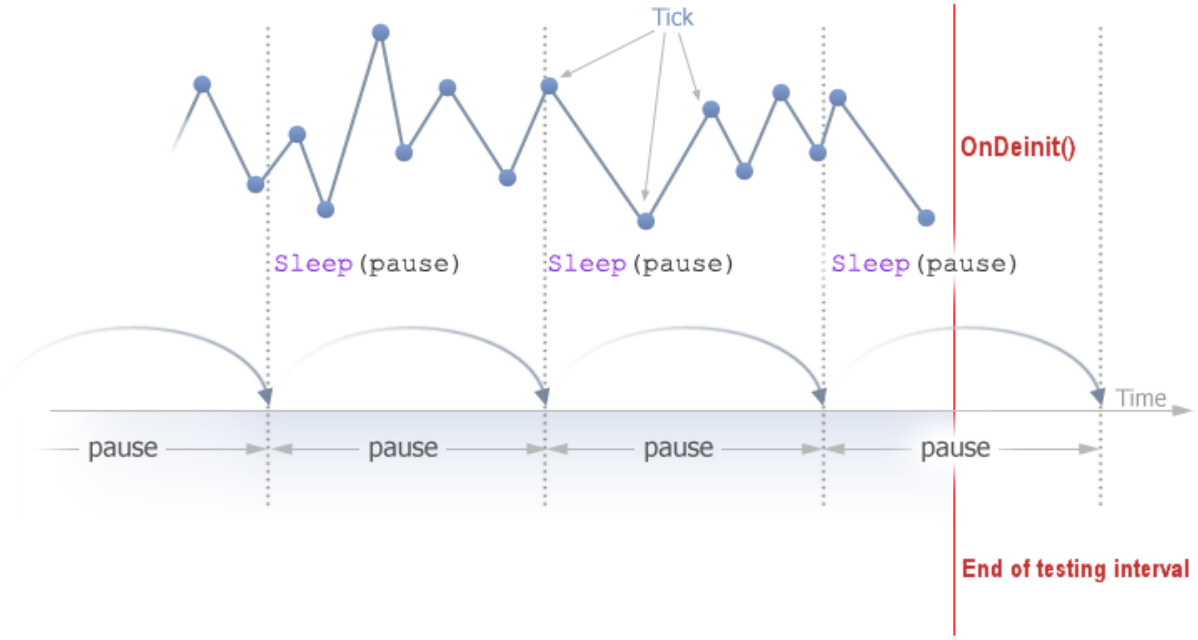
© 2000-2025, MetaQuotes Ltd.

Sleep() 함수를 사용하면 그래프 작업 시 EA 또는 스크립트가 mql5 프로그램의 실행을 잠시 중단할 수 있습니다. 이것은 요청 시 준비되지 않아 준비될 때까지 기다려야 하는 데이터를 요청할 때 유용할 수 있습니다. Sleep() 함수의 자세한 사용 예는 데이터 액세스 구성 섹션에서 확인할 수 있습니다.

테스트 프로세스는 Sleep() 호출에 의해 유지되지 않습니다. Sleep()을 호출하면, 생성된 틱이 지정된 지연 시간 내에 "재생"되어 보류 주문, 스탑 등이 트리거 될 수 있습니다. Sleep() 호출 뒤, Strategy Tester의 시뮬레이션 기간은 Sleep 함수의 매개 변수에 지정된 간격만큼 증가합니다.

Sleep() 함수를 실행한 결과, Strategy Tester의 현재 시간이 테스트 기간을 초과한 경우 "테스트 중 무한 절전 루프가 감지됨" 오류가 발생합니다. 이 오류가 발생하면 테스트 결과가 거부되지 않고 모든 계산이 전체 볼륨(거래 수, 감퇴 등)으로 수행되며 테스트 결과가 터미널로 전달됩니다.

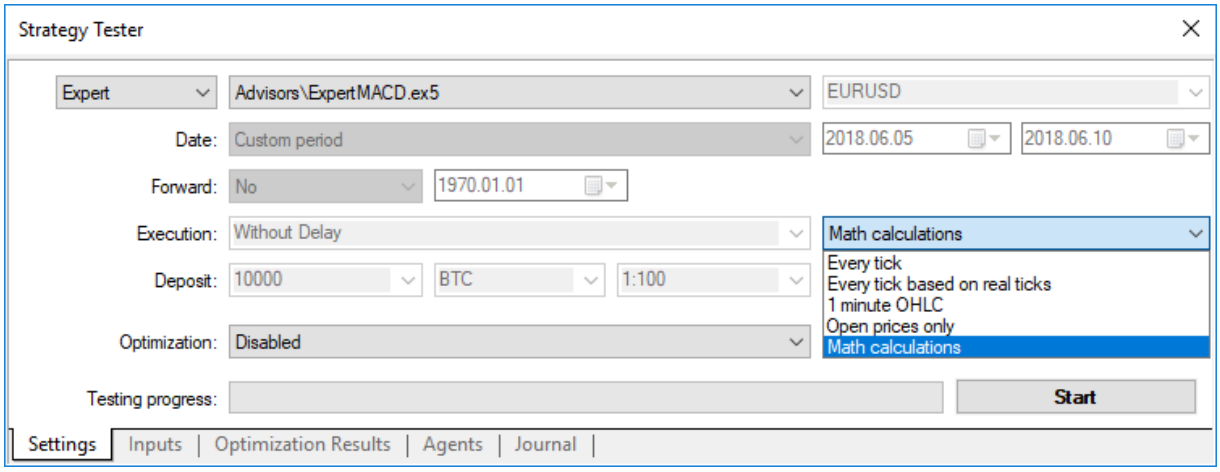
호출 후에는 테스트 시간이 테스트 간격 범위를 초과하므로 OnDeinit()에서는 Sleep() 함수가 작동하지 않습니다.



수학 계산에서 최적화 문제를 위한 전략 테스터 사용

MetaTrader 5 터미널의 테스터는 거래 전략 테스트뿐만 아니라 수학적 연산에도 사용할 수 있습니다. 이를 사용하려면 "수학 연산" 모드를 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.



이 경우 다음 세 함수만 호출이 가능합니다: OnInit(), OnTester(), OnDeinit(). "수학 연산" 모드에서는 Strategy Tester가 틱을 생성하거나 내역을 다운로드하지 않습니다.

시작 날짜를 종료 날짜보다 크게 지정한 경우에도 전략 테스터는 "수학 연산" 모드에서 작동합니다.

테스터를 사용하여 수학적 문제를 해결할 때 내역 업로드 및 틱 생성이 수행되지 않습니다.

MetaTrader 5 Strategy Tester에서 풀 수 있는 일반적 수학 문제 – 변수가 많은 함수의 극한값 찾기.

그 문제를 해결하려면 다음을 해야 합니다:

- 함수 값의 계산은 OnTester() 함수에 위치해야 합니다;
- 함수의 매개 변수는 Expert Advisor의 입력 변수로 정의해야 합니다;

EA를 컴파일하고 "Strategy Tester" 창을 엽니다. "입력 매개 변수" 탭에서 필요한 입력 변수를 선택하고 각 함수의 변수들의 시작, 중지 및 단계 값을 지정하여 매개 변수 값 집합을 정의합니다.

최적화 유형 선택 – "느리지만 완전한 알고리즘"(매개 변수 공간의 전체 검색) 또는 "빠른 유전자 기반 알고리즘". 함수의 극단을 간단하게 검색하려면 빠른 최적화를 선택하는 것이 좋지만 전체 변수 집합에 대한 값을 계산하려면 느린 최적화를 사용하는 것이 좋습니다.

"수학 연산" 모드를 선택하고 "시작" 버튼을 사용하여 최적화 절차를 실행합니다. 최적화 중 Strategy Tester는 OnTester 함수의 최대값을 검색합니다. 로컬 최소값을 찾으려면 OnTester 함수에서 계산된 함수 값의 역수를 반환합니다.

```
return(1/function_value);
```

function_value가 0이 아닌지 점검할 필요가 있습니다. 그렇지 않으면 0으로 나누는 심각한 오류를 얻을 수도 있기 때문입니다.

다른 방법이 있는데, 그것은 더 편리하고 최적화의 결과를 왜곡하지 않았고, 그것은 이 기사의 독자들에 의해 제안되었습니다:

```
return(-function_value);
```

이 옵션은 function_value가 0인지 확인할 필요가 없으며 최적화 결과 표면에서 3D-표현은 동일한 형태를 가집니다. 유일한 차이점은 원본과 비교하여 미러링된다는 것입니다.

한 예로, sink() 함수를 보면:

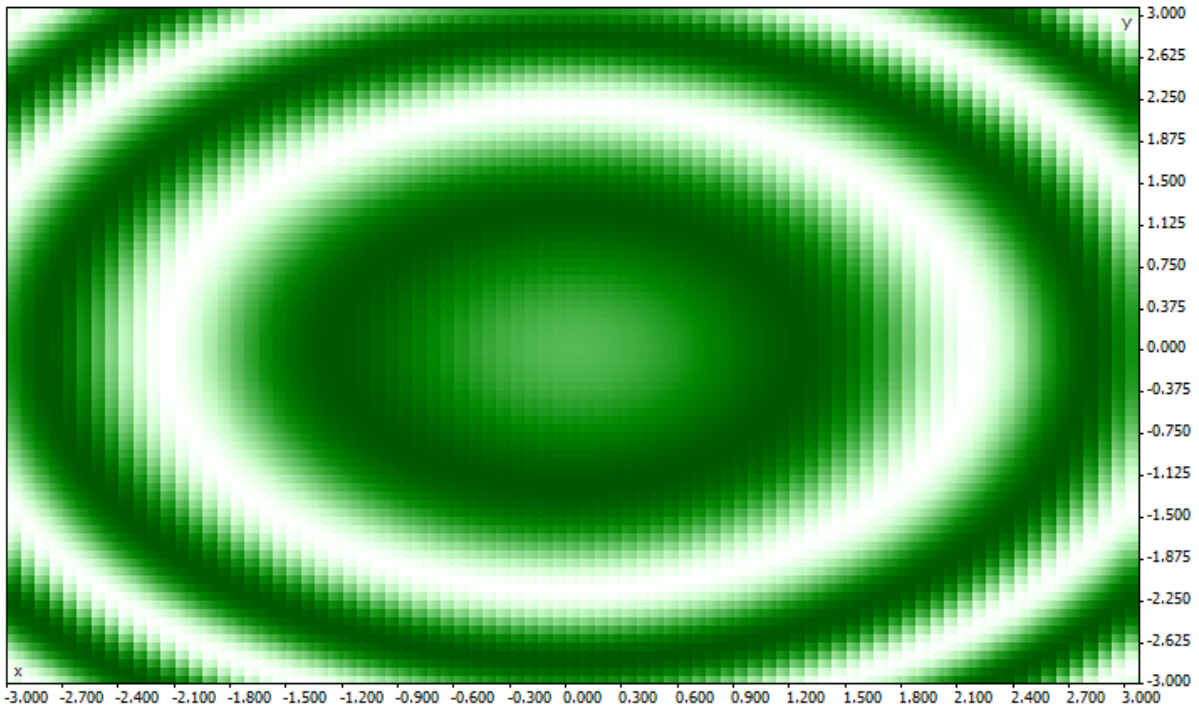
© 2000-2025, MetaQuotes Ltd.

$$sink(x,y) = \sin(x^2 + y^2)$$

이 함수의 극단을 찾기 위한 EA 코드는 OnTester()에 저장됩니다:

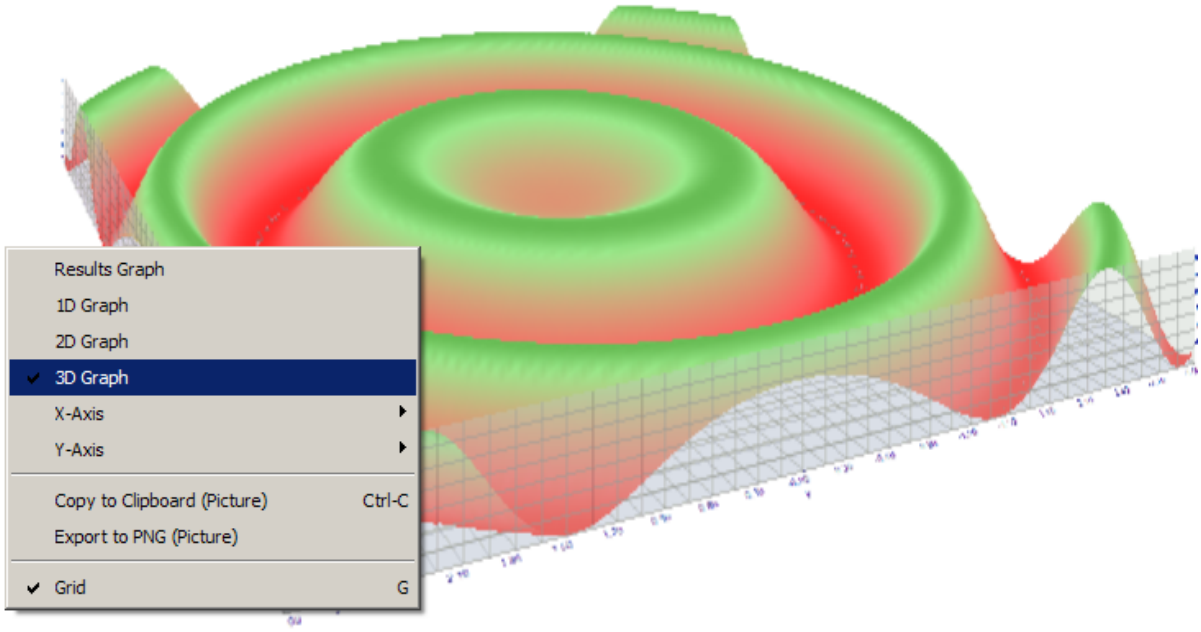
```
//+------------------------------------------------------------------+
//|                                                         Sink.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 입력 매개 변수
input double   x=-3.0; // start=-3, step=0.05, stop=3
input double   y=-3.0; // start=-3, step=0.05, stop=3
//+------------------------------------------------------------------+
//| 테스터 함수                                                      |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------+
```

최적화를 수행하고 최적화 결과를 2D 그래프 형식으로 확인합니다.



© 2000-2025, MetaQuotes Ltd.

지정된 파라미터 쌍(x, y)에 대해 값이 좋을수록 색상이 더 포화됩니다. sink() 함수의 관점에서 예상되었듯이, 그 값은 (0.0)의 중심에 동심원을 형성합니다. 3D 그래프에서 sink() 함수에 단일 글로벌 극단이 없음을 확인할 수 있습니다:



"개방 가격 전용" 모드의 막대 동기화

MetaTrader 5 클라이언트 터미널의 테스터를 통해 우리는 소위 "다중 통화" EA를 확인할 수 있습니다. 다중 통화 EA – 는 두 개 이상의 심볼을 거래하는 EA입니다.

여러 심볼에 따라 거래되는 전략 테스트는 테스터에게 몇 가지 추가 기술적 사항이 요구됩니다:

- 이러한 심볼에 대한 틱의 생성;
- 이러한 심볼에 대한 지표 값 계산;
- 이러한 심볼에 대한 마진 요구 사항의 계산;
- 모든 거래 심볼에 대해 생성된 틱 시퀀스의 동기화.

전략 테스터는 선택된 거래 모드에 따라 각 금융상품에 대해 탁 시퀀스를 생성하고 재생합니다. 동시에 다른 심볼에서 막대가 어떻게 오픈됐는지에 관계없이 각 심볼에 대한 새 막대가 오픈됩니다. 즉, 다중-통화 EA를 테스트할 때 한 금융상품의 경우 새 막대가 이미 열려 있고 다른 금융상품의 경우 열려 있지 않은 상황이 발생할 수 있습니다(그리고 자주 발생하고 있음). 따라서, 테스트 중, 모든 것은 실제와 똑같이 일어납니다.

이 테스터의 기록을 실제로 시뮬레이션하는 것은 "Every tick(모든 틱 표시)" 및 "1분 OHLC" 테스트 모드를 사용하는 한 문제를 일으키지 않습니다. 이러한 모드의 경우 다른 심볼에서 막대가 동기화될 때까지 대기할 수 있는 충분한 틱의 캔들스틱이 생성됩니다. 하지만 거래 금융상품 막대의 동기화가 필수적인 경우, 어떻게 "시가 전용" 모드에서 다중-통화 전략을 테스트할 수 있을까요? 이 모드에서 EA는 막대가 오픈된 시간에 해당하는 하나의 틱에서만 호출됩니다.

예를 들어 EA를 EURUSD에서 테스트하고 있으며 새로운 H1 캔들스틱이 EURUSD에서 오픈됐습니다. 이러한 사실을 쉽게 알 수 있습니다 – "시가 전용" 모드에서 테스트하는 동안, NewTick 이벤트는 테스트된 기간에 막대가 오픈되는 순간과 일치합니다. 하지만 EA에서 사용되는 USDJPY 심볼에 새 캔들스틱이 오픈되었다는 보장은 없습니다.

© 2000-2025, MetaQuotes Ltd.

정상적인 상황에서는 OnTick() 함수의 작업을 완료하고 다음 틱에서 USDJPY에 새로운 막대가 나타나는지 확인하는 것으로 충분합니다. 그러나 "시가 전용" 모드에서 테스트할 때는 다른 틱이 없으므로 이 모드는 다중 통화 EA 테스트에는 적합하지 않은 것으로 보일 수 있습니다. 그러나 이는 그렇지 않습니다 - MetaTrader 5의 테스터는 실제와 똑같이 동작한다는 점을 잊지 마십시오. Sleep() 함수를 사용하여 다른 심볼에 새 막대가 오픈될 때까지 기다릴 수 있습니다!

"시가 전용" 모드에서 막대를 동기화하는 예를 보여주는 EA Synchronize_Bars_Use_Sleep.mq5 코드:

```
//+------------------------------------------------------------------+
//|                                   Synchronize_Bars_Use_Sleep.mq5 |
//|                        Copyright 2011, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 입력 매개 변수
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 심볼 체크
   if(_Symbol==other_symbol)
     {
      PrintFormat("입력 매개 변수에 다른 심볼을 지정하거나 전략 테스터에서 다른 심볼을 선택해야
      //--- 강제 스탑 테스트
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 정적 변수, 마지막 막대 시간 저장에 사용됨
   static datetime last_bar_time=0;
//--- 싱크 플래그
   static bool synchonized=false;
//--- 정적 변수가 초기화되지 않은 경우
   if(last_bar_time==0)
     {
      //--- 첫 번째 호출, 바 시간 저장 및 종료
      last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("The last_bar_time variable is initialized with value %s",TimeToStri
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 차트 심볼의 마지막 막대 오픈 시간 가져오기
   datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DAT
//--- 시간이 동일하지 않을 경우
   if(curr_time!=last_bar_time)
     {
      //--- 정적 변수에 대한 바 오픈 시간 저장
      last_bar_time=curr_time;
      //--- 동기화되지 않음
      synchonized=false;
      //--- print message
      PrintFormat("심볼에 새 막대가 표시되었습니다 %s at %s",_Symbol,TimeToString(TimeCurr
     }
//--- 다른 심볼 막대의 오픈 시간
   datetime other_time;
//--- 다른 심볼의 오픈 시간이 curr_time과 같아질 때까지 루프
   while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),SE
     {
      PrintFormat("5초 대기중..");
      //--- 5초 동안 대기했다가 SeriesInfoInteger 호출 (other_symbol,Period(),SERIES_LAST
      Sleep(5000);
     }
//--- 막대가 동기화되었음
   synchonized=true;
   PrintFormat("차트 심볼의 막대 오픈 시간 %s: is %s",_Symbol,TimeToString(last_bar_time)
   PrintFormat("심볼의 막대 오픈 시간 %s: is %s",other_symbol,TimeToString(other_time));
//--- TimeCurrent()은 유용하지 않으니, TimeTradeServer()를 사용하십시오
   Print("막대가 다음에 동기화되었습니다 ",TimeToString(TimeTradeServer(),TIME_SECONDS));
  }
//+------------------------------------------------------------------+
```

EA의 마지막 줄에 동기화된 사실이 설정된 현재 시간을 표시합니다:

```
   Print("막대가 다음에 동기화되었습니다 ",TimeToString(TimeTradeServer(),TIME_SECONDS));
```

현재 시간을 표시하기 위해 TimeTradeServer() 함수를 사용했습니다, TimeCurrent()가 아니라. TimeCurrent() 함수는 마지막 틱의 시간을 반환합니다. 이 시간은 Sleep()을 사용한 후에도 변경되지 않습니다. "시가 전용" 모드에서 EA를 실행하면 막대의 동기화에 대한 메시지가 표시됩니다.

Core 1	2010.12.01 20:00:05	The bars are synchronized at 2010.12.01 20:00:05
Core 1	2010.12.01 20:00:05	Open bar time of the chart symbol USDJPY: 2010.12.01 20:00
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 20:00:00	Waiting 5 seconds..
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 16:00:05	The bars are synchronized at 2010.12.01 16:00:05

마지막 틱 도착 시간이 아니라 현재 서버 시간을 가져와야 하는 경우 TimeCurrent() 대신 TimeTradeServer() 함수를 사용하십시오.

막대를 동기화하는 또 다른 방법은 타이머를 사용하는 것입니다. 그러한 EA의 예는 본 문서에 첨부되어 있는 Synchronize_Bars_Use_OnTimer.mq5입니다.

© 2000-2025, MetaQuotes Ltd.

테스터의 IndicatorRelease() 함수

단일 테스트를 완료하면 완료된 거래와 EA에서 사용된 지표를 표시하는 금융상품 차트가 자동으로 열립니다. 이를 통해 진입점과 종료점을 시각적으로 확인하고 이를 지표 값과 비교할 수 있습니다.

> **참고:** 차트에 표시되는 지표는 테스트 완료 후 자동으로 오픈됩니다. 비록 이 지표들이 테스트된 EA에서 사용되었다 하더라도 말입니다.

그러나 경우에 따라 프로그래머는 어떤 지표가 거래 알고리즘에 관여했는지에 대한 정보를 숨기고 싶어할 수 있습니다. 예를 들어, EA의 코드는 소스 코드를 제공하지 않고 실행 파일로 대여 또는 판매됩니다. 이러한 목적으로는, IndicatorRelease() 함수가 적합합니다.

터미널이 클라이언트 터미널의 디렉토리/프로파일/템플릿에 tester.tpl이라는 이름의 템플릿을 설정하면 오픈 차트에 적용됩니다. 없으면 기본 템플릿이 적용됩니다. (default.tpl).

IndicatorRelease() 함수는 원래 더 이상 필요하지 않은 지표의 연산 부분을 해제하기 위한 것입니다. 이렇게 하면 각 틱마다 지표 계산이 필요하므로 메모리와 CPU 리소스를 모두 저장할 수 있습니다. 두 번째 목적은 한 번의 테스트 실행 후 테스트 차트에 지표를 표시하는 것을 금지하는 것입니다.

테스트 후 차트에 지표가 표시되지 않도록 하려면 IndicatorRelease() 호출을 핸들러 OnDeinit()의 지표 핸들을 사용해서 해야 합니다. OnDeinit() 함수는 항상 테스트 차트가 표시되기 전에 호출됩니다.

```
//+------------------------------------------------------------------+
//| Expert 초기화 취소 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden) Print("IndicatorRelease() successfully completed");
   else Print("IndicatorRelease() returned false. Error code ",GetLastError());
  }
```

차트에 지표가 표시되지 않도록 하려면 단일 테스트를 완료한 후 핸들러 OnDeinit()의 IndicatorRelease() 함수를 사용하십시오.

테스터의 이벤트 핸들링

EA에 OnTick() 핸들러가 있는 것은 MetaTrader 5 테스터의 기간별 데이터에 대한 테스트를 받기 위한 필수 사항이 아닙니다. EA의 ti가 다음 함수-핸들러 중 하나 이상을 포함하기에 충분합니다:

- OnTick() - 새 틱 도착의 이벤트 핸들러;
- OnTrade() - 트레이딩 이벤트 핸들러;
- OnTimer() - 타이머로부터의 시그널 도착에 대한 이벤트 핸들러;
- OnChartEvent() - 클라이언트 이벤트에 대한 핸들러.

EA에서 테스트할 때, OnChartEvent() 함수를 사용하여 커스텀 이벤트를 처리할 수 있지만 지표에서는 이 함수를 테스터에서 호출할 수 없습니다. 지표에 OnChartEvent() 이벤트 핸들러가 있고 이 지표가 테스트된 EA에서 사용되는 경우에도 지표 자체는 커스텀 이벤트를 수신하지 않습니다.

© 2000-2025, MetaQuotes Ltd.

테스트 중에 지표는 EventChartCustom() 함수를 사용하여 커스텀 이벤트를 생성할 수 있으며, EA는 OnChartEvent()에서 이 이벤트를 처리할 수 있습니다.

이러한 이벤트 외에도, 테스트 및 최적화 프로세스와 관련된 특수 이벤트가 전략 테스터에서 생성됩니다.

- 테스터 – 이 이벤트는 Expert Advisor에서 내역 데이터에 대한 테스트를 완료한 후 생성됩니다. 테스터 이벤트는 OnTester() 함수를 사용하여 처리됩니다. 이 함수는 Expert Advisor를 테스트할 때만 사용할 수 있으며, 주로 입력 파라미터의 유전 최적화를 위한 Custom max 기준으로 사용되는 값을 계산하기 위한 것입니다.
- TesterInit – 이 이벤트는 첫 번째 패스 전에 전략 테스터에서 최적화를 시작할 때 생성됩니다. TesterInit 이벤트는 OnTesterInit() 함수를 사용하여 처리됩니다. 최적화가 시작되는 동안 이 핸들러를 사용하는 Expert Advisor는 테스터에 지정된 심볼과 주기가 있는 별도의 터미널 차트에 자동으로 로드되고 TesterInit 이벤트를 수신합니다. 이 함수는 추가적인 최적화 결과 처리를 위해 최적화를 시작하기 전에 Expert Advisor를 시작하는 데 사용됩니다.
- TesterPass – 이 이벤트는 새 데이터 프레임이 수신될 때 생성됩니다. TesterPass 이벤트는 OnTesterPass() 함수를 사용하여 처리됩니다. 이 핸들러를 사용하는 Expert Advisor는 테스트를 위해 지정된 심볼/주기가 있는 별도의 터미널 차트에 자동으로 로드되며, 최적화 중에 프레임이 수신되면 TesterPass 이벤트를 수신합니다. 이 함수는 최적화 결과의 완료를 기다리지 않고 "즉각" 동적으로 처리하는 데 사용됩니다. 프레임은 FrameAdd() 함수를 사용하여 추가됩니다 – 이는OnTester() 핸들러에서 단일 패스가 끝난 후 호출할 수 있는 함수입니다.
- TesterDeinit – 이 이벤트는 Expert Advisor 최적화 종료 후 전략 테스터에서 생성됩니다. TesterDeinit 이벤트는 OnTesterDeinit() 함수를 사용하여 처리합니다. 이 핸들러를 사용하는 Expert Advisor는 최적화가 시작될 때 자동으로 차트에 로드되고 완료되면 TesterDeinit를 받게 됩니다. 이 함수는 모든 최적화 결과의 최종 처리에 사용됩니다.

테스팅 에이전트

MetaTrader 5 클라이언트 터미널에서의 테스트는 테스트 에이전트를 사용하여 수행합니다. 로컬 에이전트가 자동으로 생성되고 실행됩니다. 기본 로컬 에이전트 수는 시스템의 코어 수에 해당합니다.

각 테스트 에이전트에는 클라이언트 터미널과 관련이 없는 고유한 글로벌 변수 복사본이 있습니다 터미널 자체는 태스크를 로컬 및 원격 에이전트로 배포하는 발송 관리자입니다. 지정된 매개 변수를 사용하여 EA 테스트에 대한 작업을 실행한 후 에이전트는 결과를 터미널로 반환합니다. 한 번의 테스트에서는 에이전트가 하나만 사용됩니다.

에이전트는 터미널에서 수신한 내역을 기기 이름으로 별도의 폴더에 저장하므로 EURUSD의 기록은 EURUSD라는 폴더에 저장됩니다. 또한 금융상품의 이력은 소스에 따라 구분됩니다. 내역을 저장하는 구조는 다음과 같습니다:

```
tester_catalog\Agent-IPaddress-Port\bases\name_source\history\symbol_name
```

예를 들어 MetaQuotes-Demo 서버의 EURUSD 내역은 folder tester_catalog₩Agent-127.0.0.1-3000₩bases₩MetaQuotes-Demo₩EURUSD 폴더에 저장할 수 있습니다.

로컬 에이전트는 테스트가 완료된 후 대기 모드로 전환되어 다음 호출을 시작하는 데 시간을 낭비하지 않도록 5분 동안 다음 작업을 대기합니다. 대기 기간이 끝난 후에만 로컬 에이전트가 종료되고 CPU 메모리에서 언로드됩니다.

클라이언트 터미널 종료와 함께 사용자 측에서 테스트를 조기에 완료한 경우, 모든 로컬 에이전트는 즉시 작업을 중지하고 메모리에서 언로드됩니다.

© 2000-2025, MetaQuotes Ltd.

터미널과 에이전트 간의 데이터 교환

테스트를 실행하면 클라이언트 터미널은 에이전트에게 다음과 같은 여러 매개 변수 블록을 보낼 준비를 합니다:

- 테스트를 위한 입력 파라미터(시뮬레이션 모드, 시험 간격, 기구, 최적화 기준 등)
- "Market Watch"에서 선택한 심볼 목록
- 테스트 심볼의 명세(계약규모, 스탑로스 및 이익실현 설정의 마켓으로부터의 허용된 마진 등)
- 테스트할 Expert Advisor와 해당 입력 매개 변수의 값
- 추가 파일(라이브러리, 지표, 데이터 파일)에 대한 정보 –# property tester_ ...)

tester_indicator	string	"indicator_name.ex5"의 포맷의 커스텀 지표명. 테스트가 필요한 지표는 해당 파라미터가 문자열 상수를 통해 설정된 경우 iCustom() 함수 호출 시 자동으로 정의됩니다. 다른 모든 경우(지표 이름을 설정하는 파라미터에서 IndicatorCreate() 함수 사용 또는 문자열 비상수 사용)에는 이 속성이 필요합니다
tester_file	string	확장명이 표시된 테스터의 파일 이름으로는 큰따옴표로(문자열 상수로써) 표시됩니다. 지정한 파일이 테스터로 전달됩니다. 필요한 경우 테스트할 입력 파일을 항상 지정해야 합니다.
tester_library	string	확장명이 큰따옴표로 표시된 라이브러리 이름. 라이브러리는 dll 또는 ex5 확장자를 가질 수 있습니다. 테스트가 필요한 라이브러리는 자동으로 정의됩니다. 그러나 커스텀 지표에서 라이브러리를 사용하는 경우 이 속성이 필요합니다.

각 매개 변수 블록에 대해 MD5-해시 형식의 디지털 지문이 생성되어 에이전트로 전송됩니다. MD5 해시는 각 세트에 대해 고유하며, MD5 해시의 볼륨은 계산된 정보의 양보다 훨씬 작습니다.

에이전트는 블록 해시를 수신하여 이미 있는 블록과 비교합니다. 지정된 매개 변수 블록의 지문이 에이전트에 없거나 수신된 해시가 기존 해시와 다른 경우 에이전트가 이 매개 변수 블록을 요청합니다. 그러면 터미널과 에이전트 간의 트래픽이 줄어듭니다.

테스트 후 에이전트는 "테스트 결과" 및 "최적화 결과" 탭에 표시된 모든 실행 결과(수령 이익, 거래 수, 샤프 계수, OnTester() 함수 결과 등)를 터미널에 반환합니다.

최적화하는 동안 터미널은 테스트 작업을 작은 패키지로 에이전트에 제공하며, 각 패키지에는 여러 작업이 포함되어 있습니다(각 태스크는 입력 매개 변수 세트를 사용한 단일 테스트를 의미합니다). 그러면 터미널과 에이전트 간의 교환 시간이 줄어듭니다.

에이전트는 보안상의 이유로 터미널(EA, 지표, 라이브러리 등)에서 가져온 EX5 파일을 하드 디스크에 기록하지 않으므로 에이전트가 실행 중인 컴퓨터는 전송된 데이터를 사용할 수 없습니다. DLL을 포함한 다른 모든 파일은 샌드박스에 기록됩니다. 원격 에이전트에서는 DLL을 사용하여 EA를 테스트할 수 없습니다.

테스트 결과는 터미널에 의해 특수 결과 캐시(결과 캐시)에 추가되어 필요할 때 신속하게 액세스할 수 있습니다. 각 매개 변수 세트에 대해 터미널은 재실행을 방지하기 위해 결과 캐시에서 이전 실행의 이미 사용 가능한 결과를 검색합니다. 이러한 매개 변수 집합이 있는 결과를 찾을 수 없는 경우 에이전트에 테스트를 수행하는 작업이 부여됩니다.

터미널과 에이전트 간의 모든 트래픽이 암호화됩니다.

© 2000-2025, MetaQuotes Ltd.

틱은 네트워크를 통해 전송되지 않고 테스트 에이전트에서 생성됩니다.

모든 클라이언트 터미널의 공유 폴더 사용

모든 테스트 에이전트는 서로 분리되고 클라이언트 터미널에서 분리됩니다. 각 에이전트에는 로그를 기록하는 고유한 폴더가 있습니다. 또한 에이전트 테스트 중 모든 파일 작업은 **agent_name/MQL5/Files**폴더에서 수행됩니다. 그러나 파일을 여는 동안 플래그 FILE_COMMON를 지정하는 경우 모든 클라이언트 터미널의 공유 폴더를 통해 로컬 에이전트와 클라이언트 터미널 간의 상호 작용을 구현할 수 있습니다:

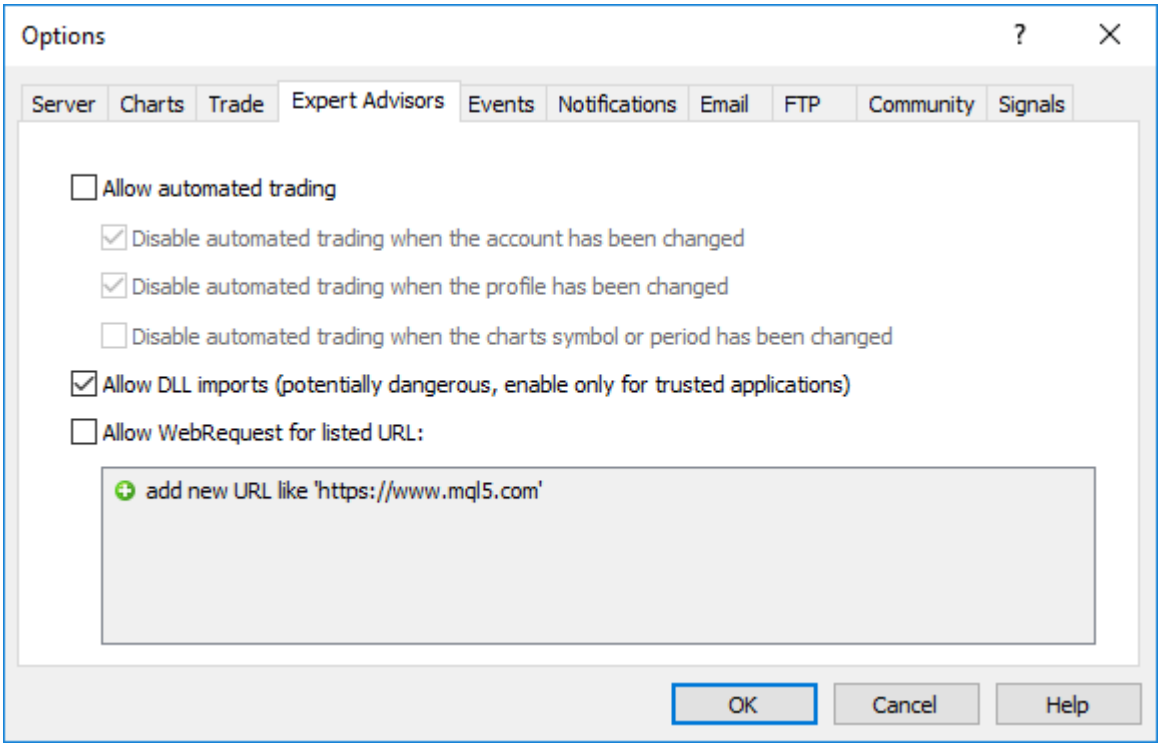
```
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 모든 클라이언트 터미널의 공유 폴더
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- 이 폴더의 이름을 빼냅니다
   PrintFormat("클라이언트 터미널의 공유 폴더에 있는 파일 오픈 %s", common_folder);
//--- 공유 폴더에서 파일 오픈 (FILE_COMMON 플래그로 표시됨)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
  ... 추가 조치
//---
   return(INIT_SUCCEEDED);
  }
```

DLLs 사용

최적화 속도를 높이기 위해 로컬 에이전트 뿐 아니라 원격 에이전트도 사용할 수 있습니다. 이 경우 원격 에이전트에는 몇 가지 제한이 있습니다. 먼저 원격 에이전트는 로그에서 Print() 함수의 실행 결과, 즉 포지션 오픈 및 클로즈에 대한 메시지를 표시하지 않습니다. 잘못 작성된 EA가 원격 에이전트가 작업 중인 시스템을 메시지와 함께 추적하지 못하도록 로그에 최소한의 정보가 표시됩니다.

두 번째 제한 사항 -EA 테스트 시 DLL 사용 금지. 보안상의 이유로 원격 에이전트에서는 DLL 호출이 절대 금지됩니다. 로컬 에이전트에서 테스트된 EA의 DLL 호출은 적절한 "Allow import DLL" 권한으로만 허용됩니다.

© 2000-2025, MetaQuotes Ltd.



参考: 허용된 DLL 호출이 필요한 타사 EA(스크립트, 지표)를 사용할 때는 터미널 설정에서 이 옵션을 허용할 때 발생할 수 있는 위험을 알고 있어야 합니다. EA가 사용되는 방법에 관계없이 테스트 또는 차트 실행에 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

미리 정의된 변수

각 실행가능한 mql5 프로그램에는 mql5 프로그램(Expert Advisor, 스크립트 또는 커스텀 지표)이 시작되는 시점까지 현재 가격 차트의 상태를 반영하는 사전 정의된 변수 집합이 지원됩니다.

미리 정의된 변수의 값은 mql5 프로그램이 시작되기 전에 클라이언트 터미널에 의해 설정됩니다. 미리 정의된 변수는 일정하므로 mql5 프로그램에서 변경할 수 없습니다. 예외적으로, 특수 변수 _LastError 가 있는데, 이 변수는 ResetLastError 함수로 0으로 재설정될 수 있습니다.

변수	값
_AppliedTo	_AppliedTo 변수를 사용하면 지표 계산에 사용되는 데이터 유형을 확인할 수 있습니다
_Digits	(십진법)자릿수
_Point	견적 통화의 현재 심볼 포인트 크기
_LastError	최근 에러 코드
_Period	현재 차트의 타임프레임
_RandomSeed	의사 난수 생성기의 현재 상태
_StopFlag	프로그램 스탑 플래그
_Symbol	현재 차트의 심볼명
_UninitReason	미초기화 근거 코드
_IsX64	_IsX64 변수를 사용하면 MQL5 응용 프로그램이 실행 중인 터미널의 비트 버전을 확인할 수 있습니다.

라이브러리에서 미리 정의된 변수를 정의할 수 없습니다. 라이브러리는 이 라이브러리가 호출되는 프로그램에 정의된 변수를 사용합니다.

© 2000-2025, MetaQuotes Ltd.

int _AppliedTo

_AppliedTo 변수를 사용하면 지표 계산에 사용되는 데이터 유형을 확인할 수 있습니다.:

데이터 유형	의미	지표 계산에 사용되는 데이터에 대한 설명.
—	0	이 지표는 두 번째 OnCalculate() 호출 양식을 사용합니다 – 계산을 위한 데이터가 특정 버퍼 또는 데이터 배열에 의해 지정되지 않았습니다.
Close	1	종가
Open	2	시가
High	3	고가
Low	4	저가
중위가 (HL/2)	5	Median price = (High+Low)/2
표준가 (HLC/3)	6	Typical price = (High+Low+Close)/3
가중가 (HLCC/4)	7	Weighted price = (Open+High+Low+Close)/4
이전 지표 데이터	8	이 지표보다 먼저 차트에서 시작된 지표의 데이터
첫 지표 데이터	9	차트에서 먼저 시작된 지표의 데이터
지표 핸들	10+	지표 핸들을 사용하여 iCustom() 함수에 전달된 지표 데이터. _AppliedTo 값에 지표 핸들이 포함되어 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 매핑
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
// 지표 계산에 사용되는 데이터 유형 가져오기
   Print("_AppliedTo=",_AppliedTo);
   Print(getIndicatorDataDescription(_AppliedTo));
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 지표 계산에 사용된 데이터 설명                                   |
//+------------------------------------------------------------------+
string getIndicatorDataDescription(int data_id)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   string descr="";
   switch(data_id)
     {
      case(0):descr="첫 번째 유형의 OnCalculate() - 데이터 버퍼 없음";
         break;
      case(1):descr="지표가 종가 기준으로 계산됨.";
         break;
      case(2):descr="지표가 시가 기준으로 계산됨";
         break;
      case(3):descr="지표가 고가 기준으로 계산됨";
         break;
      case(4):descr="지표가 저가 기준으로 계산됨";
         break;
      case(5):descr="지표가 중위가(HL/2) 기준으로 계산됨";
         break;
      case(6):descr="지표가 표준가(HLC/3) 기준으로 계산됨";
         break;
      case(7):descr="지표가 가중가(HLCC/4) 기준으로 계산됨";
         break;
      case(8):descr="지표가 이전 지표의 데이터를 기준으로 계산됨";
         break;
      case(9):descr="지표가 첫번째 지표의 데이터를 기준으로 계산됨";
         break;
      default: descr="지표가 핸들을 지닌 다음 지표의 데이터를 기준으로 계산됨="+string(data_
         break;
     }
//---
   return descr;
  }
```

참고 항목

ENUM_APPLIED_PRICE

© 2000-2025, MetaQuotes Ltd.

int _Digits

_Digits 변수는 현재 차트의 심볼에 대한 가격 정확도를 정의하는 소수점 뒤의 자릿수를 저장합니다.

또한 Digits() 함수도 사용할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

double _Point

_Point 変数は 見積 通貨に ある 現在 シンボル の ポイント サイズ を 含む.

また Point() 関数 を 使用 する こと も できる.

© 2000-2025, MetaQuotes Ltd.

int _LastError

_LastError 변수에는 mql5-프로그램 실행 중에 발생한 마지막 에러의 코드가 포함되어 있습니다. ResetLastError()를 통해 값을 0으로 재설정할 수 있습니다.

마지막 오류의 코드를 얻으려면 GetLastError() 함수를 사용할 수도 있습니다.

© 2000-2025, MetaQuotes Ltd.

ENUM_TIMEFRAMES _Period

_Period 변수는 현재 차트의 타임프레임 값을 포함합니다.

또한 Period() 함수를 사용할 수도 있습니다.

참고 항목

PeriodSeconds, 차트 타임프레임, 날짜 시간, 객체 가시성

© 2000-2025, MetaQuotes Ltd.

_RandomSeed

의사 임의 정수를 생성할 때 현재 상태를 저장하는 변수입니다. _RandomSeed 는 MathRand()를 호출할 때 값을 변경합니다. 필요한 초기 조건을 설정하려면 MathSrand() 를 사용합니다.

x 난수는 MathRand() 함수로 수신되며 각 호출에서 다음 방법으로 계산됩니다:

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

참고 항목

MathRand(), MathSrand(), Integer types

© 2000-2025, MetaQuotes Ltd.

int _StopFlag

_StopFlag 변수에는 정상적인 작동 중에 0인 mql5 프로그램 중지 플래그가 포함되어 있습니다. 클라이언트 터미널이 프로그램을 중지하려고 할 경우 변수가 0이 아닌 값으로 설정됩니다.

_StopFlag의 상태를 확인하려면 IsStopped() 함수를 사용하실 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

string _Symbol

_Symbol 변수에는 현재 차트의 심볼명이 포함되어 있습니다.

또한 Symbol() 함수를 사용하실 수도 있습니다.

© 2000-2025, MetaQuotes Ltd.

int _UninitReason

_UninitReason 변수에는 프로그램 미초기화 근거 코드가 포함되어 있습니다.

일반적으로 이 코드는 UninitializeReason() 함수에서 얻습니다.

© 2000-2025, MetaQuotes Ltd.

int _IsX64

_IsX64 변수를 사용하면 MQL5 응용 프로그램이 실행 중인 터미널의 비트 버전을 확인할 수 있습니다: 32비트 터미널의 경우 _IsX64=0, 64비트 터미널의 경우 _IsX64!=0.

또한, 함수 TerminalInfoInteger(TERMINAL_X64)가 사용될 수 있습니다.

예:

```
// 프로그램이 실행 중인 터미널 확인
   Print("_IsX64=",_IsX64);
   if(_IsX64)
      Print("프로그램 ",__FILE__,"이 64비트 터미널에서 실행 중입니다.");
   else
      Print("프로그램 ",__FILE__,"이 32비트 터미널에서 실행 중입니다.");
   Print("TerminalInfoInteger(TERMINAL_X64)=",TerminalInfoInteger(TERMINAL_X64));
```

참고 항목

MQLInfoInteger, 임포팅 함수 (#import)

© 2000-2025, MetaQuotes Ltd.

공통 함수

특수 그룹에 포함되지 않은 범용 기능이 여기에 나열.

기능	액션
Alert	별도의 창에 메시지를 표시
CheckPointer	개체 포인터의 유형을 반환
Comment	차트의 왼쪽 상단 코너에 코멘트를 출력
CryptEncode	지정된 방법을 사용하여 배열에서 데이터를 변환
CryptDecode	배열에서 데이터의 역변환을 수행
DebugBreak	디버깅의 중지포인트 프로그램
ExpertRemove	엑스퍼트 어드바이저를 중지하고 차트에서 언로드
GetPointer	반환하는 개체포인터
GetTickCount	시스템이 시작된 후 경과된 밀리초 수를 반환
GetTickCount64	시스템이 시작된 후 경과된 밀리초 수를 반환
GetMicrosecondCount	MQL5 프로그램을 시작한 후 경과한 시간(마이크로초)을 반환
MessageBox	메시지 상자를 만들고 표시 및 관리
PeriodSeconds	기간 내 초 수를 반환
PlaySound	사운드 파일을 재생
Print	로그에 메시지 표시
PrintFormat	미리 설정된 형식에 따라 로그 파일 심볼 및 값 세트를 포맷 및 인쇄
ResetLastError	사전 결정된 변수 _LastError 값을 0으로 설정
ResourceCreate	데이터 세트 기반으로 이미지 리소스 생성
ResourceFree	삭제: 동적으로 생성된 리소스 (해당 리소스에 할당된 메모리 사용 가능)
ResourceReadImage	다음의 그래픽 리소스에서 데이터 읽기: ResourceCreate() 기능으로 생성 또는 컴파일 중EX5 파일에 저장
ResourceSave	지정된 파일에 리소스를 저장
SetUserError	사전 정의된 변수 _LastError를 ERR_USER_ERROR_FIRST + user_error와 동일한 값으로 설정
SetReturnError	작업을 완료할 때 터미널 프로세스를 반환하는 코드를 설정
Sleep	지정된 간격 내에 엑스퍼트 어드바이저 또는 스크립트의 실행을 일시 중단
TerminalClose	터미널이 작동을 완료하도록 명령
TesterHideIndicators	EA에 사용되는 지표의 표시/숨김 모드를 설정

© 2000-2025, MetaQuotes Ltd.

기능	액션
TesterStatistics	테스트 결과를 기반으로 계산된 지정된 통계의 값을 반환
TesterStop	다음을 할 때 프로그램 작동 완료 명령을 제공테스트
TesterDeposit	테스트 중에 예금 자금을 에뮬레이트 함. 일부 자금 관리 시스템에 사용할 수 있습니다
TesterWithdrawal	테스트 과정에서 자금 인출 운영 에뮬레이트 함
TranslateKey	가상 키 코드로 유니코드 문자를 반환
ZeroMemory	참조로 전달된 변수를 재설정 변수는 컨스트럭터가 있는 클래스 및 구조를 제외한 모든 유형일 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

Alert

메시지를 별도의 창에 표시

```
void  Alert(
   argument,      // 첫 번째 값
   ...            // 다른 값
   );
```

매개변수

논의

[in] 쉼표로 구분된 값. 정보 출력을 여려 줄로 분할하기 위해 라인 피드 문자 "₩n" 또는 "₩r₩n"을 사용할 수 있습니다. 매개변수의 수는 64개를 초과할 수 없습니다.

값 반환

반환 값 없음.

주의

배열을 Alert() 함수에 전달 할 수 없습니다. 배열은 요소별로 출력되어야 합니다. double 유형의 데이터는 소수점 아래 8자리 숫자까지 출력되고, float 유형의 데이터는 소수점 아래 5자리 숫자로 표시됩니다. 실제 숫자를 다른 정밀도 또는 과학적 형식으로 출력하려면 DoubleToString() 함수를 사용.

bool 유형의 데이터는 "true" 또는 "false" 문자열로 출력됩니다. 날짜는 YYYY.MM.DD HH:MI:SS 로 출력됩니다. 날짜를 다른 형식으로 표시하려면 TimeToString() 함수를 사용. 색상 유형의 대이터는 색상이 색상 집합에 있는 경우 R, G, B 문자열 또는 색상 이름으로 출력.

전략 테스터에서 Alert() 기능이 작동하지 않습니다.

예:

```
//--- enums
enum ENUM_INTERSECT_DIRECTION
  {
   INTERSECT_DIRECTION_NONE= 0,  // 교차 없음
   INTERSECT_DIRECTION_UP  = 1,  // 상방 교차
   INTERSECT_DIRECTION_DOWN=-1,  // 하방 교차
  };

//--- 입력 매개변수
input    uint                InpPeriod = 10;            // MA 기간
input    int                 InpShift  = 0;             // MA 쉬프트
input    ENUM_MA_METHOD      InpMethod = MODE_SMA;      // MA 메서드
input    ENUM_APPLIED_PRICE  InpPrice  = PRICE_CLOSE;   // MA 적용 가격

//--- 글로벌 변수
int      ExtMaHandle;
int      ExtMaPeriod;
double   ExtData[2];
```

© 2000-2025, MetaQuotes Ltd.

```
MqlRates ExtRates[2];

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 입력 매개변수에 0을 지정하면 이동 평균을 계산하는 기간은 기본값(10)과 같습니다.r
   ExtMaPeriod=int(InpPeriod<1 ? 10 : InpPeriod);
//--- 지정된 매개변수를 사용하여 이동 평균 지표에 대한 핸들을 생성합니다.
   ExtMaHandle=iMA(Symbol(),PERIOD_CURRENT,ExtMaPeriod,InpShift,InpMethod,InpPrice);
   ResetLastError();
   if(ExtMaHandle==INVALID_HANDLE)
     {
      PrintFormat("Failed to create iMA() handle. Error code: %d",GetLastError());
      return(INIT_FAILED);
     }

//--- 마지막 가격의 업데이트 시간을 가져옵니다.
   datetime tick_time=TickTime();
//--- 마지막 두 캔들에서 이동 평균 데이터와 가격 데이터를 가져옵니다.
   if(GetData(ExtMaHandle,ExtData,ExtRates) && tick_time!=0)
     {
      //--- 가격이 MA보다 높은 경우
      if(ExtRates[1].close>ExtData[1])
        {
         //--- 메시지 텍스트 작성 및 경고 표시
         string message=StringFormat("Bar time: %s. The price is above the moving aver
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         /*
         Result:
         Alert: Bar time: 2024.02.16 18:00. The price is above the moving average at 2
         */
        }
      else
        {
         //--- 가격이 MA보다 낮은 경우
         if(ExtRates[1].close<ExtData[1])
           {
            //--- 메시지 텍스트 작성 및 경고 표시
            string message=StringFormat("Bar time: %s. The price is below the moving a
            Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SE
            /*
            Result:
            Alert: Bar time: 2024.02.16 19:00. The price is below the moving average a
            */
           }
         else
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            //--- 메시지 텍스트 작성 및 경고 표시
            string message=StringFormat("Bar time: %s. The price and moving average ar
            Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SE
            /*
            Result:
            Alert: Bar time: 2024.02.16 20:00. The price and moving average are equal
            */
           }
        }
     }

//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
   ResetLastError();
//--- 마지막 두 캔들에서 이동 평균 데이터와 가격 데이터를 가져옵니다.
   if(!GetData(ExtMaHandle,ExtData,ExtRates))
      return;
//--- 현재 캔들의 이동평균을 교차하는 가격의 방향을 구합니다.
   ENUM_INTERSECT_DIRECTION intersect=GetIntersectDirection(ExtData,ExtRates);

//--- 이전 메시지를 저장하기 위한 변수
   static string message_prev="";

//--- 가격이 현재 캔들의 이동 평균을 상방 교차한 경우
   if(intersect==INTERSECT_DIRECTION_UP)
     {
      //--- 교차가 발생한 틱 시간을 가져옵니다.
      datetime tick_time=TickTime();
      if(tick_time==0)
         return;
      //--- 메세지 텍스트를 생성
      string message=StringFormat("Bar time: %s. The price crossed the MA from bottom
      //--- 이전 메시지가 현재 메시지와 같지 않으면 메시지 및 틱 시간과 함께 경고를 표시합니다.
      if(message!=message_prev)
        {
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         message_prev=message;
         /*
         Result:\
         Alert: Bar time: 2024.02.16 09:00. The price crossed the MA from bottom to to
         */
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 가격이 현재 바의 이동 평균을 하향 교차한 경우
   if(intersect==INTERSECT_DIRECTION_DOWN)
     {
      //--- 교차가 발생한 틱 시간을 가져옵니다.
      datetime tick_time=TickTime();
      if(tick_time==0)
         return;
      //--- 메세지 텍스트를 생성
      string message=StringFormat("Bar time: %s. The price crossed the MA from top to
      //--- 이전 메시지가 현재 메시지와 같지 않으면 메시지 및 틱 시간과 함께 경고를 표시합니다.
      if(message!=message_prev)
        {
         Alert(message+" at "+TimeToString(tick_time,TIME_DATE|TIME_MINUTES|TIME_SECON
         message_prev=message;
         /*
         Result:\
         Alert: Bar time: 2024.02.16 10:00. The price crossed the MA from top to botto
         */
        }
     }
  }
//+------------------------------------------------------------------+
//| 가격을 확인하고 평균 데이터를 배열로 이동                                |
//+------------------------------------------------------------------+
bool GetData(int handle,double &ma_data[],MqlRates &price_data[])
  {
   ResetLastError();
//--- 마지막 두 바에서 이동 평균 데이터를 가져옵니다.
   if(CopyBuffer(handle,0,0,2,ma_data)!=2)
     {
      PrintFormat("CopyBuffer() failed. Error code: %d",GetLastError());
      return(false);
     }
//--- 마지막 두 바의 가격 데이터를 가져옵니다.
   if(CopyRates(Symbol(),PERIOD_CURRENT,0,2,price_data)!=2)
     {
      PrintFormat("CopyRates() failed. Error code: %d",GetLastError());
      return(false);
     }

   return(true);
  }
//+------------------------------------------------------------------+
//| 이동평균을 교차하는 가격의 방향을 반환합니다.     |
//+------------------------------------------------------------------+
ENUM_INTERSECT_DIRECTION GetIntersectDirection(double &ma_data[],MqlRates &price_data
  {
   double ma0=ma_data[1];
```

© 2000-2025, MetaQuotes Ltd.

```
   double ma1=ma_data[0];
   double close0=price_data[1].close;
   double close1=price_data[0].close;

   if(close1<=ma1 && close0>ma0)
      return(INTERSECT_DIRECTION_UP);
   else
     {
      if(close1>=ma1 && close0<ma0)
         return(INTERSECT_DIRECTION_DOWN);
      else
         return(INTERSECT_DIRECTION_NONE);
     }
  }
//+------------------------------------------------------------------+
//| 틱 시간을 초 단위로 반환합니다.
//+------------------------------------------------------------------+
datetime TickTime()
  {
   MqlTick tick={};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      PrintFormat("SymbolInfoTick() failed. Error code: %d",GetLastError());
      return(0);
     }

   return(tick.time);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckPointer

함수는 개체 포인터의 유형을 반환.

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject      // 개체 포인터
   );
```

매개변수

anyobject

[in] 개체 포인터.

값을 반환

ENUM_POINTER_TYPE 열거에서 값을 반환.

주의

잘못된 포인터를 호출 시도로 인해 프로그램의 심각한 종료가 발생합니다. 그렇기 때문에 포인터를 사용하기 전에 CheckPointer 함수를 호출해야 합니다. 다음과 같은 경우 포인터가 잘못될 수 있습니다:

- 포인터가 NULL과 같음;
- 개체가 삭제 연산자를 사용하여 삭제.

이 기능은 포인터 유효성 검사에 사용할 수 있습니다. 값이 0이 아닌 경우 포인터가 액세스에 사용될 수 있음을 보증합니다.

포인터의 유효성을 빠르게 확인하려면 "!" 연산자를 사용할 수도 있습니다. (예) CheckPointer 함수의 묵시적 호출을 통해 이를 확인합니다.

예를 들어:

```
//+------------------------------------------------------------------+
//| 요소를 삭제하여 목록 삭제                                          |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- 루프에서 작업하기 위한 서비스 포인터
   CItem* item;
//--- 루프를 통해 동적 포인터를 삭제를 시도
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dynamic object ",item.Identifier()," to be deleted");
         delete (item);
        }
      else Print("비동적 개체 ",item.Identifier()," 삭제될 수 없습니다");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
  }
```

추가 참조

개체 포인터, 개체 포인터 확인, 개체 삭제 연산자 삭제

© 2000-2025, MetaQuotes Ltd.

Comment

이 함수는 차트의 왼쪽 상단 모서리에 사용자가 정의한 코멘트를 출력.

```
void  Comment(
   argument,      // 첫 번째 값
   ...            // 다음 값
   );
```

매개변수

...

[in] 숨표로 구분된 임의의 값. 출력 정보를 여러 줄로 구분하기 위해 줄바꿈 기호 "₩n" 또는 "₩r₩n" 이 사용됩니다. 매개변수 수는 64개를 초과할 수 없습니다. 입력 코멘트의 총 길이(보이지 않는 심볼 포함)는 2045자를 초과할 수 없습니다(출력 중에 초과 심볼이 삭제됨).

값 반환

반환 값 없음

주의

배열을 Comment() 기능에 전달할 수 없습니다. 배열은 요소별로 입력해야 합니다.

double 유형의 데이터는 소수점 아래 최대 16자리까지의 정확도로 출력되며, 어떤 표기법이 더 압축되느냐에 따라 전통적인 형식이나 과학적 형식으로 출력될 수 있습니다. float 유형의 데이터는 소수점 아래 5자리 숫자로 출력됩니다. 다른 정확도 또는 미리 정의된 형식으로 실제 숫자를 출력하려면 DoubleToString() 함수를 사용합니다.

bool 유형의 데이터는 "true" 또는 "false" 문자열로 출력됩니다. 날짜는 YYYY.MM.DD HH:MI:SS 로 표시됩니다. 날짜를 다른 형식으로 표시하려면 TimeToString() 기능을 사용. 색상 유형 데이터는 R, G, B string으로 출력되거나 색상 세트에 이 색상이 있는 경우 색상 이름으로 출력됩니다.

Comment() 기능은 전략 테스터의 최적화 중에 작동하지 않습니다.

예를 들어:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- 세 줄의 출력 값
   Comment(StringFormat("Show prices\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,Spread)
  }
```

추가 참조

ChartSetString, ChartGetString

© 2000-2025, MetaQuotes Ltd.

CryptEncode

지정된 방법을 사용하여 배열의 데이터를 변환.

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,        // 방법
   const uchar&        data[],        // 소스 배열
   const uchar&        key[],         // 키
   uchar&              result[]       // 대상 배열
   );
```

매개변수

방법

[in] 데이터 변환 방법. ENUM_CRYPT_METHOD 열거 값 중 하나일 수 있습니다.

data[]

[in] 소스 배열.

key[]

[in] 키 배열.

result[]

[out] 대상 배열.

값 반환

대상 배열의 바이트 양 또는 오류의 경우 0. 오류에 대한 정보를 얻으려면 GetLastError() 기능을 호출.

예를 들어:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- 확인
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);
//--- HEX string으로 변환
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string text="갈색 여우는 게으른 개를 뛰어넘는다";
```

© 2000-2025, MetaQuotes Ltd.

```
   string keystr="ABCDEFG";
   uchar src[],dst[],key[];
//--- 키 준비
   StringToCharArray(keystr,key);
//--- 소스 배열 src[]로 텍스트 복사
   StringToCharArray(text,src);
//--- 초기 데이터 인쇄
   PrintFormat("초기 데이터: size=%d, string='%s'",ArraySize(src),CharArrayToString(src
//--- key[]에서 DES 56 비트 키를 사용하여 src[] 암호화
   int res=CryptEncode(CRYPT_DES,src,key,dst);
//--- 오류 확인
   if(res>0)
     {
      //--- 암호화 데이터 인쇄
      PrintFormat("인코드된 데이터: size=%d %s",res,ArrayToHex(dst));
      //--- dst[]를 src[]로 해독
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- 오류 확인
      if(res>0)
        {
         //--- 해독된 데이터 인쇄
         PrintFormat("해독된 데이터: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("CryptDecode에 오류. Error code=",GetLastError());
     }
   else
      Print("CryptEncode에 오류. Error code=",GetLastError());
  }
```

추가 참조

문자열 기능, CryptDecode()

© 2000-2025, MetaQuotes Ltd.

CryptDecode

CryptEncode()에 의 해 변환된 배열에서 데이터의 역 변환을 수행.

```
int  CryptEncode(
   ENUM_CRYPT_METHOD   method,        // 메서드
   const uchar&        data[],        // 소스 배열
   const uchar&        key[],         // 키
   uchar&              result[]       // 대상 배열
   );
```

매개변수

메서드

[in] 데이터 변환 메서드. ENUM_CRYPT_METHOD 열거 값 중 하나일 수 있습니다.

data[]

[in] 소스 배열.

key[]

[in] 키 배열.

result[]

[out] 대상 배열.

값 반환

대상 배열의 바이트 양 또는 오류의 경우 0. 오류에 대한 정보를 얻으려면 GetLastError() 기능을 호출.

예:

```
input string InpKey = "ABCDEFG";  // 암호화 키

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string text="The quick brown fox jumps over the lazy dog";
   uchar src[],dst[],key[];

//--- 암호화 키 준비
   StringToCharArray(InpKey,key);
//--- src[] 소스 배열 준비
   StringToCharArray(text,src);
//--- 소스 데이터 표시
   PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(s

//--- key[] 56비트 키를 사용하여 DES 메서드를 사용하여 src[] 배열을 암호화합니다.
   int res=CryptEncode(CRYPT_DES,src,key,dst);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 암호화 결과 확인
   if(res>0)
     {
      //--- 암호화된 데이터 표시
      PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
      //--- key[] 56비트 키를 사용하여 DES 메서드를 사용하여 dst[] 배열 데이터를 해독합니다.
      res=CryptDecode(CRYPT_DES,dst,key,src);
      //--- 결과 확인
      if(res>0)
        {
         //--- 복호화된 데이터 표시
         PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToSt
        }
      else
         Print("CryptDecode failed. Error: ",GetLastError());
     }
   else
      Print("CryptEncode failed. Error: ",GetLastError());
  }
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";

//--- 크기 확인
   if(count<0 || count>ArraySize(arr))
      count=ArraySize(arr);

//--- 16진수 문자열로 변환
   for(int i=0; i<count; i++)
      res+=StringFormat("%.2X",arr[i]);

   return(res);
  }
```

추가 참조

문자열 기, CryptEncode()

© 2000-2025, MetaQuotes Ltd.

DebugBreak

디버깅할 때 프로그램 중단점.

```
void  DebugBreak();
```

값 반환

반환 값 없음.

주의

MQL5 프로그램의 실행은 프로그램이 디버깅 모드에서 시작된 경우에만 중단됨. 이 기능은 변수의 값을 보거나 추가 단계 실행을 위해 사용할 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- F5를 사용하여 파일을 컴파일합니다.
//--- 디버깅 모드에서 i == j 이면 DebugBreak() 문자열에서 중지합니다.
   for(int i=0,j=20; i<20; i++,j--)
     {
      if(i==j)
         DebugBreak();
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

ExpertRemove

이 함수는 엑스퍼트 어드바이저를 중지하고 차트에서 언로드.

```
void  ExpertRemove();
```

값 반환

반환 값 없음.

주의

ExpertRemove()를 호출할 때 엑스퍼트 어드바이저가 즉시 중지되는 것은 아니며, EA 작업을 중지하기 위한 플래그만 설정됩니다. 즉, 다음 이벤트가 처리되지 않고, OnDeinit()이 호출되고 엑스퍼트 어드바이저가 언로드되어 차트에서 제거됩니다.

ExpertRemove()를 OnInit() 핸들러 내의 전략테스터에서 호출하면 현재의 매개변수 세트에 대한 테스트가 취소됩니다. 이러한 완료는 초기화 오류로 간주됨.

EA를 전략 테스터에서 ExpertRemove() 를 호출할 때성공적으로 초기화 한 후라면 OnDeinit() 및 OnTester()의 호출로 정상적으로 테스트가 완료됩니다. 이 경우 전체 거래 통계와 최적화 기준 값을 구한다.

예를 들어:

```
//+------------------------------------------------------------------+
//|                                              Test_ExpertRemove.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20;// EA 언로드 전 틱의 수
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__," reason code = ",reason);
//--- "비움" 코멘트
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
   static int tick_counter=0;
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   tick_counter++;
   Comment("\nBefore unloading expert advisor ",__FILE__," 좌측",
           (ticks_to_close-tick_counter)," 틱");
//--- 전
   if(tick_counter>=ticks_to_close)
     {
      ExpertRemove();
      Print(TimeCurrent(),": ",__FUNCTION__," 엑스퍼트 어드바이저는 언로드됩니다");
     }
   Print("tick_counter =",tick_counter);
//---
  }
//+------------------------------------------------------------------+
```

추가 참조

프로그램 실행, 클라이언트 터미널 이벤트

© 2000-2025, MetaQuotes Ltd.

GetPointer

이 함수는 개체 포인터를 반환.

```
void*  GetPointer(
   any_class anyobject      // 모든 클래스의 개체
   );
```

매개변수

anyobject

[in] 모든 클래스의 개체.

값 반환

함수는 개체 포인터를 반환.

주의

클래스 개체에만 포인터가 있습니다. 구조 및 단순 유형 변수의 인스턴스에는 포인터를 사용할 수 없습니다. 클래스 객체는 new() 연산자를 사용하여 생성되지 않지만, 예를 들어 개체 배열에서 자동으로 생성되지만, 포인터는 여전히 존재합니다. 그러나 이 포인터는 자동 타입 POINTER_AUTOMATIC이므로, delete() 연산자를 적용할 수 없습니다. 이와는 별도로, 유형 포인터는 POINTER_DYNAMIC 유형의 동적 포인터와 다르지 않습니다.

구조 유형 및 단순 유형의 변수에는 포인터가 없으므로 GetPointer() 함수를 적용할 수 없습니다. 포인터를 함수 인수로 전달하는 것도 금지됩니다. 이 모든 경우에 컴파일러는 오류를 통지합니다.

잘못된 포인터를 호출 시도는 프로그램의 심각한 종료를 야기합니다. 그렇기 때문에 포인터를 사용하기 전에 CheckPointer()기능을 호출해야 합니다. 다음과 같은 경우 포인터가 잘못될 수 있습니다:

- 포인터가 NULL과 같음;
- 개체가 삭제 연산자를 사용하여 삭제.

포인터의 유효성을 검사하는 데 사용할 수 있습니다. 값이 0이 아닌 경우 포인터가 액세스에 사용될 수 있음을 보장합니다.

예를 들어:

```
//+------------------------------------------------------------------+
//|                                            Check_GetPointer.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+------------------------------------------------------------------+
//| 목록 요소 구현 클래스                                    |
//+------------------------------------------------------------------+
class CItem
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   int               m_id;
   string            m_comment;
   CItem*            m_next;
public:
                     CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                    ~CItem() { Print("Destructor of ",m_id,
                                     (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                                     "dynamic":"non-dynamic"); }
   void              Initialize(int id,string comm) { m_id=id; m_comment=comm; }
   void              PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
   int               Identifier() { return(m_id); }
   CItem*            Next() {return(m_next); }
   void              Next(CItem *item) { m_next=item; }
  };
//+------------------------------------------------------------------+
//| 목록의 가장 간단한 클래스                                        |
//+------------------------------------------------------------------+
class CMyList
  {
   CItem*            m_items;
public:
                     CMyList() { m_items=NULL; }
                    ~CMyList() { Destroy(); }
    bool             InsertToBegin(CItem* item);
    void             Destroy();
  };
//+------------------------------------------------------------------+
//| 시작 부분에 목록 요소를 삽입                                     |
//+------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| 요소를 삭제하여 목록 삭제                                        |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- 루프에서 작업할 수 있는 서비스 포인터
   CItem* item;
//--- 루프를 통과하여 동적 포인터를 삭제 시도
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
```

© 2000-2025, MetaQuotes Ltd.

```
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("동적 개체 ",item.Identifier()," 삭제됨");
         delete (item);
        }
      else Print("비동적 개체 ",item.Identifier()," 삭제될 수 없습니다");
     }
//---
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- 동적 개체 포인터를 만들고 목록에 추가
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- 목록에 자동 포인터를 추가
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"automatic");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- 목록에 동적 개체 포인터를 하나 더 추가
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dynamic");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- 모든 목록 요소를 삭제
   list.Destroy();
//--- 모든 목록 요소는 스크립트가 끝난 후에 삭제
//--- 터미널의 전문가 탭을 참조
  }
```

© 2000-2025, MetaQuotes Ltd.

추가 참조

개체 포인터, 개체 포인터 확인, 개체 삭제 연산자 삭제

© 2000-2025, MetaQuotes Ltd.

GetTickCount

GetTickCount() 기능은 시스템이 시작된 후 경과된 밀리초 수를 반환.

```
uint  GetTickCount();
```

값 반환

uint 유형.

주의

카운터는 시스템 타이머의 제한에 의해 제한됩니다. 시간은 부호 없는 정수로 저장되므로 컴퓨터가 중단없이 작동하면 49.7일마다 초과 채워집니다.

예를 들어:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------+
//| 40 피보나치 수 계산 시간 측정 스크립트                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 초기값 기억
   uint start=GetTickCount();
//--- 피보나치 수열의 다음 숫자를 가져오기 위한 변수
   long fib=0;
//--- 루프에서 피보나치 수열에서 지정된 수의 양을 계산
   for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- 소요시간(밀리초) 가져오기
   uint time=GetTickCount()-start;
//--- 엑스퍼트 저널에 메시지 출력
   PrintFormat("%d 첫 번째 피보나치 수 계산에서 %d ms가 소요",MAX_SIZE,time);
//--- 스크립트 완료됨
   return;
  }
//+------------------------------------------------------------------+
//| 수열 숫자로 피보나치 수를 가져오는 함수                                |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- 피보나치 수열의 첫 번째 구성
   if(n<2) return(1);
//--- 다른 모든 구성은 다음 공식으로 계산됨
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

추가 참조

날짜 및 시간, EventSetMillisecondTimer, GetTickCount64, GetMicrosecondCount

© 2000-2025, MetaQuotes Ltd.

GetTickCount64

GetTickCount64() 함수는 시스템이 시작된 후 경과된 밀리초 수를 반환.

```
ulong  GetTickCount64();
```

값 반환

ulong 유형 값.

주의

카운터는 일반적으로 10~16밀리초의 정밀도로 결과를 반환하는 시스템 타이머의 정확도로 제한됩니다. 컴퓨터 작업을 계속할 경우, GetTickCount가 49.7일마다uint 유형의 컨텐츠를 오버플로우 하는 것과 달리, GetTickCount64()는 무제한 컴퓨터 작동 시간에 사용할 수 있으며 오버플로우의 대상이 되지 않습니다.

예:

```
#define MAX_SIZE 40

//+------------------------------------------------------------------+
//| 40개의 Fibo 숫자를 계산하는 시간을 측정하는 스크립트                |
//+------------------------------------------------------------------+
void OnStart()
  {
   long fib_array[MAX_SIZE];

//--- 처음의 값을 저장
   ulong start=GetTickCount64();
//--- Fibo 시리즈에서 주어진 수의 숫자를 계산하는 루프
   for(int i=0;i<MAX_SIZE;i++)
      fib_array[i]=TestFibo(i);
//--- 소요된 시간을 밀리초 단위로 가져옵니다.
   ulong time=GetTickCount64()-start;

//--- EA 저널에 오류 메시지를 표시합니다.
   ArrayPrint(fib_array);
   PrintFormat("Calculating the first %d Fibonacci numbers took %I64u ms",MAX_SIZE,tim
  }
//+------------------------------------------------------------------+
//| 일련번호로 Fibo 번호를 얻는 함수                                  |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- Fibo 시리즈의 첫 번째 멤버
   if(n<2)
      return(1);
//--- 모든 후속 멤버는 이 방정식을 사용하여 계산됩니다.
   return(TestFibo(n-2)+TestFibo(n-1));
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

추가 참조

날짜 및 시간, EventSetMillisecondTimer, GetTickCount, GetMicrosecondCount

© 2000-2025, MetaQuotes Ltd.

GetMicrosecondCount

GetMicrosecondCount() 기능은 MQL5-프로그램 시작 후 경과된 마이크로초 수를 반환.

```
ulong  GetMicrosecondCount();
```

값 반환

ulong 유형 값.

예를 들어:

```
//+------------------------------------------------------------------+
//| 테스트 기능                                                      |
//+------------------------------------------------------------------+
void Test()
  {
   int    res_int=0;
   double res_double=0;
//---
   for(int i=0;i<10000;i++)
     {
      res_int+=i*i;
      res_int++;
      res_double+=i*i;
      res_double++;
     }
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- 측정 횟수
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //--- 측정 유형 선택
         if(n==0) ui=GetTickCount();
         else     ul=GetMicrosecondCount();
         //--- 코드 실행
         Test();
         //--- 측정 결과 추가(종류에 따라 다름)
         if(n==0) ui_res+=GetTickCount()-ui;
```

© 2000-2025, MetaQuotes Ltd.

```
         else     ul_res+=GetMicrosecondCount()-ul;
        }
      //--- 최소 및 최대 두 측정 시간을 계산
      if(ui_min>ui_res) ui_min=ui_res;
      if(ui_max<ui_res) ui_max=ui_res;
      if(ul_min>ul_res) ul_min=ul_res;
      if(ul_max<ul_res) ul_max=ul_res;
     }
//---
   Print("GetTickCount error(msec): ",ui_max-ui_min);
   Print("GetMicrosecondCount error(msec): ",DoubleToString((ul_max-ul_min)/1000.0,2))
  }
```

추가 참조

Date and Time, GetTickCount, GetTickCount64

© 2000-2025, MetaQuotes Ltd.

MessageBox

메시지 상자를 만들고 표시하고 관리합니다. 메시지 상자에는 메시지와 헤더, 미리 정의된 기호 및 명령 버튼의 조합이 포함됩니다.

```
int  MessageBox(
   string  text,              // 메시지 텍스트
   string  caption=NULL,      // 상자 헤더
   int     flags=0            // 상자에서 버튼 세트를 정의
   );
```

매개변수

text

[in] 출력할 메시지를 포함하는 텍스트

caption=NULL

[in] 상자 헤더에 표시할 선택적 텍스트. 매개변수가 비어 있으면, 상자 헤더에 엑스퍼트 어드바이저 이름이 표시됩니다.

flags=0

[in] 메시지 상자의 모양 및 동작을 정의하는 선택적 플래그. 플래그는 특별한 플래그 그룹 조합일 수 있습니다.

값 반환

I기능이 성공적으로 수행되면 MessageBox() 반환 코드 값 중 하나입니다.

주의

MessageBox()를 호출하면 사용자의 응답을 기다리는 동안실행 스레드가 일시 중단되므로 사용자 지정 지표에서 이 기능을 사용할 수 없습니다. 각 기호에 대한 모든 지표가 단일 스레드에서 실행되기 때문에, 이러한 서스펜션은 이 기호에 대한 모든 시간프레임에 대한 모든 차트의 작동을 불가능하게 만듭니다.

MessageBox() 기능은전략 테스터에서 작동하지 않습니다.

예:

```
//+------------------------------------------------------------------+
//| EA가 추가 작업을 요청하는 MessageBox 창을 표시합니다.                |
//| 일련의 수익성 없는 거래가 지정된 수에 도달하면                       |
//| 지정된 캔들 수만큼 기다렸다가 MessageBox를 표시합니다.               |
//| 작업을 계속해 달라는 요청과 함께.                                   |
//| 확인하려면 여러 포지션을 수동으로 열고 닫아야 합니다.               |
//| EA가 통제하지 않기 때문에 손실이 발생합니다.                        |
//| 단순화를 위해 매직 넘버로 자체 "위치"를 지정합니다.                 |
//+------------------------------------------------------------------+

//--- 입력 매개변수
input uint InpMaxLossDeals      =  3;    // 최대 손실 거래
```

© 2000-2025, MetaQuotes Ltd.

```
input uint InpInactivityNumBars =  5;     // EA 비활성 막대 수

//--- 글로벌 변수
bool     ExtFirstStart=true;              // 첫 번째 실행 플래그
bool     ExtFlag=true;                    // EA 작동을 허용하는 플래그
uint     ExtNumLoss;                      // 연속적으로 수익성 없는 거래 횟수
datetime ExtTimeLastLoss;                 // 손실 포지션을 청산하기 위한 마지막 거래 시간

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 연속으로 손실이 발생한 거래 수와 포지션을 청산하기 위한 마지막 거래 시간을 가져옵니다.
   ExtNumLoss=GetNumLosingTradesInRow(ExtTimeLastLoss);

   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   Comment("");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 시리즈의 마지막 청산 손실 포지션 이후 얼마나 많은 바가 지나갔는지 확인합니다.
   int bars_remaining=iBarShift(Symbol(),PERIOD_CURRENT,ExtTimeLastLoss);

//--- 만약 처음 실행인 경우
   if(ExtFirstStart)
     {
      //--- 일련의 수익성 없는 포지션 이후에 지정된 수의 바가 이미 지나간 경우 EA 작업 플래그를
      if(bars_remaining>(int)InpInactivityNumBars)
         ExtFlag=true;
      ExtFirstStart=false;
     }

//--- EA 작업 플래그가 비활성화된 경우
   if(!ExtFlag)
     {
      Comment(StringFormat("The advisor is stopped for %d bars. Num Loss positions: %u
                           (InpInactivityNumBars-bars_remaining),ExtNumLoss,TimeToStrin
      //--- 일련의 수익성 없는 포지션 이후에 지정된 수의 바가 지나간 경우
      if(bars_remaining>(int)InpInactivityNumBars)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- 지정된 텍스트와 창 제목을 사용하여 MessageBox 창을 표시합니다.
          //--- 요청 창에는 두 개의 예/아니요 버튼과 물음표가 있는 아이콘이 있습니다.
          //--- 기본적으로 예 버튼이 선택되어 있습니다.
          string mb_text="The specified number of bars of EA inactivity have passed.\n
          string mb_caption="Please note";
          int    mb_id=MessageBox(mb_text,mb_caption,MB_YESNO|MB_ICONQUESTION|MB_DEFBUT
          //--- MessageBox의 반환 코드가 Yes 버튼을 누른 경우 EA 작업 플래그를 설정합니다.
          if(mb_id==IDYES)
            {
             ExtFlag=true;
             return;
            }
         }
       //--- EA 작업 플래그가 비활성화되었습니다. OnTick()을 종료합니다.
       return;
      }

//---  EA 작업 플래그가 설정되었습니다. EA는 아래 코드에서 제공하는 대로 작동합니다.
   Comment(StringFormat("The advisor is working. Num Loss positions: %u, Time last los
           ExtNumLoss,TimeToString(ExtTimeLastLoss,TIME_DATE|TIME_MINUTES|TIME_SECONDS
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction& trans,
                        const MqlTradeRequest& request,
                        const MqlTradeResult& result)
  {
//--- 거래 유형이 내역에 거래를 추가하는 경우
   if(trans.type==TRADE_TRANSACTION_DEAL_ADD)
     {
      //--- 딜 티켓을 받고 티켓별로 목록에서 딜을 선택하세요
      ulong deal_ticket=trans.deal;
      if(HistoryDealSelect(deal_ticket))
        {
         //--- 시장 청산 거래인 경우 연속 손실 거래 수와 마지막 거래 시간을 가져옵니다.
         ENUM_DEAL_ENTRY entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(deal_ticket,DEAI
         if(entry==DEAL_ENTRY_OUT || entry==DEAL_ENTRY_INOUT || entry==DEAL_ENTRY_OUT_
            ExtNumLoss=GetNumLosingTradesInRow(ExtTimeLastLoss);
        }
     }

//--- 연속된 손실 거래 수가 지정된 값보다 크고 EA 작업 플래그가 활성화된 경우
   if(ExtNumLoss>=InpMaxLossDeals && ExtFlag)
     {
      //--- 지정된 텍스트와 창 제목이 있는 MessageBox 창 표시
      //--- 요청 창에는 두 개의 예/아니요 버튼과 느낌표가 있는 아이콘이 있습니다.
      //--- 기본적으로 아니요 버튼이 선택되어 있습니다.
```

© 2000-2025, MetaQuotes Ltd.

```
      string mb_text="The number of losing trades has reached the specified maximum. T
      string mb_caption="Attention!";
      int    mb_id=MessageBox(mb_text,mb_caption,MB_YESNO|MB_ICONQUESTION|MB_DEFBUTTON
      //--- MessageBox의 반환 코드가 No 버튼을 누른 경우 EA 작업 플래그를 비활성화합니다.
      if(mb_id==IDNO)
         ExtFlag=false;
     }
  }
//+------------------------------------------------------------------+
//| 연속적으로 수익성이 없는 거래 횟수를 반환합니다.                          |
//| 손실 포지션을 청산하기 위한 마지막 거래 시간                             |
//+------------------------------------------------------------------+
uint GetNumLosingTradesInRow(datetime &time_last_deal)
  {
//--- 전체 기록을 선택합니다.
   if(!HistorySelect(0,TimeCurrent()))
      return(0);

//--- 루프의 과거 거래 목록을 통해 다음 거래 티켓을 가져옵니다.
   uint res=0;
   uint total=HistoryDealsTotal();
   for(int i=(int)total-1; i>=0; i--)
     {
      ulong deal_ticket=HistoryDealGetTicket(i);
      if(deal_ticket>0)
        {
         //--- 거래가 포지션 종료를 위한 것이 아닌 경우 다음 거래로 이동
         ENUM_DEAL_ENTRY entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(deal_ticket,DEAL
         if(entry!=DEAL_ENTRY_OUT && entry!=DEAL_ENTRY_OUT_BY && entry!=DEAL_ENTRY_INO
            continue;
         //--- 포지션 청산 결과에 이익이 있으면 루프를 중단합니다.
         if(!IsClosePositionWithLoss(deal_ticket))
            break;
         //--- 연속 손실 거래 카운터 증가
         res++;
         //--- 최대 거래 시간을 변수에 씁니다(마지막 항목 찾기)
         datetime deal_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
         if(deal_time>time_last_deal)
            time_last_deal=deal_time;
        }
     }

//--- 연속 손실 횟수를 반환합니다.
   return(res);
  }
//+------------------------------------------------------------------+
//| 손실이 있는 포지션 청산 플래그 반환                                    |
//+------------------------------------------------------------------+
bool IsClosePositionWithLoss(const ulong deal_ticket)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 거래 이익에 영향을 미치는 속성 값을 가져옵니다.
   double profit=HistoryDealGetDouble(deal_ticket,DEAL_PROFIT);
   double comission=HistoryDealGetDouble(deal_ticket,DEAL_COMMISSION);
   double swap=HistoryDealGetDouble(deal_ticket,DEAL_SWAP);
   double fee=HistoryDealGetDouble(deal_ticket,DEAL_FEE);

//--- 수신된 속성의 총 값이 음수임을 나타내는 플래그를 반환합니다.
   return(profit+comission+swap+fee<0);
  }
```

© 2000-2025, MetaQuotes Ltd.

PeriodSeconds

이 함수는 기간(초)을 반환합니다.

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // 차트 기간
   );
```

매개변수

period=PERIOD_CURRENT

[in] 열거 ENUM_TIMEFRAMES의 차트 기간 값. 매개변수를 지정하지 않으면 프로그램이 실행되는 현재 차트 기간의 초 수를 반환합니다.

값 반환

선택된 기간의 초 수.

예:

```
//--- 입력 매개변수
input ENUM_TIMEFRAMES InpPeriod1 = PERIOD_CURRENT;    // First Period
input ENUM_TIMEFRAMES InpPeriod2 = PERIOD_M1;         // Second Period

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- InpPeriod1 및 InpPeriod2 차트 기간의 초의 수를 가져옵니다.
   int sec1=PeriodSeconds(InpPeriod1);
   int sec2=PeriodSeconds(InpPeriod2);
//--- 수신된 값을 로그에 표시합니다.
   PrintFormat("Seconds in period %s: %lu, in period %s: %lu",TimeframeDescription(Inp
//--- 차트 기간이 InpPeriod1인 바에 포함된 InpPeriod2 차트 기간의 바의 수를 계산합니다.
   int res=sec1/sec2;
   if(res==0)
      res=1;
//--- 얻은 값을 로그에 표시합니다.
   PrintFormat("One bar %s contains %d bars %s",TimeframeDescription(InpPeriod1),res,T
   /*
   Result:
   Seconds in period M5: 300, in period M1: 60
   One bar M5 contains 5 bars M1
   */
  }
//+------------------------------------------------------------------+
//| 차트 주기명 반환                                                 |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
string TimeframeDescription(const ENUM_TIMEFRAMES period)
  {
   return(StringSubstr(EnumToString(period==PERIOD_CURRENT ? Period() : period), 7));
  }
```

추가 참조

_Period, 차트 시간프레임, 날짜 및 시간, 객체의 가시성

© 2000-2025, MetaQuotes Ltd.

PlaySound

사운드 파일을 재생.

```
bool  PlaySound(
   string  filename      // 파일 이름
   );
```

매개변수

파일명

[in] 사운드 파일의 경로. filename=NULL 이면, 재생이 중지됩니다.

값 반환

발견하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

주의

파일은 terminal_directory₩Sounds 또는 그 하위 디렉토리에 있어야 합니다. WAV 파일만 재생됩니다.

NULL 매개변수를 사용하여 PlaySound()를 호출하면 재생이 중지됩니다.

PlaySound() function는 전략 테스터에서 기능이 작동하지 않음.

예:

```
#include <Trade\Trade.mqh>
#define MAGIC (123)

//--- 입력 매개변수
input string          InpFileNameOK  = "ok.wav";               // success audio file
input string          InpFileNameErr = "timeout.wav";          // error audio file
input ENUM_ORDER_TYPE InpOrderType   = ORDER_TYPE_BUY_LIMIT;   // order type
input double          InpLots        = 0.1;                    // lots

//--- 글로벌 변수
CTrade ExtTrade;

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 심볼 설정에 따라 실행에 따른 매직 및 주문 유형을 설정합니다.
   ExtTrade.SetExpertMagicNumber(MAGIC);
   ExtTrade.SetTypeFillingBySymbol(Symbol());
//--- 소리 재생으로 주문을 하거나 포지션에 진입하는 함수를 호출합니다.
   OrderSendWithAudio();
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 이 함수는 주문을 하거나 포지션을 오픈합니다                                |
//| 성공 또는 오류 소리를 재생합니다.                                         |
//+------------------------------------------------------------------------+
void OrderSendWithAudio(void)
  {
   bool    res=true;
   MqlTick tick= {};

   ResetLastError();
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      PlaySound(InpFileNameErr);
      return;
     }
//--- 서버에 요청을 보냅니다
   switch(InpOrderType)
     {
      case ORDER_TYPE_BUY :
         res=ExtTrade.Buy(InpLots);
         break;
      case ORDER_TYPE_BUY_LIMIT :
         res=ExtTrade.BuyLimit(InpLots,NormalizeDouble(tick.ask-100*Point(),Digits()))
         break;
      case ORDER_TYPE_BUY_STOP :
         res=ExtTrade.BuyStop(InpLots,NormalizeDouble(tick.ask+100*Point(),Digits()));
         break;
      case ORDER_TYPE_SELL :
         res=ExtTrade.Sell(InpLots);
         break;
      case ORDER_TYPE_SELL_LIMIT :
         res=ExtTrade.SellLimit(InpLots,NormalizeDouble(tick.bid+100*Point(),Digits())
         break;
      case ORDER_TYPE_SELL_STOP :
         res=ExtTrade.SellStop(InpLots,NormalizeDouble(tick.bid-100*Point(),Digits()))
         break;
      default :
         res=false;
     }
   if(!res)
      Print("Error ",GetLastError());
   Print(ExtTrade.ResultRetcodeDescription());

//--- 요청이 수락되면 ok.wav 사운드를 재생합니다.
   if(ExtTrade.ResultRetcode()==TRADE_RETCODE_DONE)
      PlaySound(InpFileNameOK);
   else
      PlaySound(InpFileNameErr);
  }
```

© 2000-2025, MetaQuotes Ltd.

추가 참조

리소스

© 2000-2025, MetaQuotes Ltd.

Print

엑스퍼트 어드바이저 로그에 메시지를 입력. 매개변수는 어떤 유형이든 될 수 있습니다.

```
void  Print(
   argument,      // 첫 번째 값
   ...            // 다음 값
   );
```

매개변수

...

[in] 쉼표로 구분된 값. 매개변수의 수는 64개를 초과할 수 없습니다.

주의

배열을 Print() 함수에 전달할 수 없습니다. 배열은 요소별 입력이어야 합니다.

double 유형 데이터는 소수점 이후 최대 16자리의 정확도로 표시되며, 어떤 항목이 더 압축되느냐에 따라 기존 형식이나 과학적 형식으로 출력할 수 있습니다. float 유형의 데이터는 소수점 아래 5자리 숫자로 출력됩니다. 다른 정확도 또는 미리 정의된 형식으로 실제 숫자를 출력하려면 PrintFormat() 기능을 사용합니다.

bool 유형의 데이터는 "true" 또는 "false" 라인으로 출력됩니다. 날짜는 YYYY.MM.DD HH:MI:SS 로 표시됩니다. 데이터를 다른 형식으로 표시하려면 TimeToString()을 사용. 색상 유형 데이터는 색상 집합에 있는 경우, R,G,B 라인 또는 색상 이름으로 반환됨.

전략 테스터의 최적화 중에는 Print() 기능이 작동하지 않습니다.

예를 들어:

```
void OnStart()
  {
//--- Print()를 사용하여 DBL_MAX를 출력하며, 이는 PrintFormat("%%.16G",DBL_MAX)와 동일합니다
   Print("---- DBL_MAX의 모습 -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//--- PrintFormat()을 사용하여 DBL_MAX 번호를 출력
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- 엑스퍼트 저널로 출력
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- float 출력 방법 보기
   float c=(float)M_PI; // 목표 유형에 명시적으로 캐스팅해야 합니다
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- 실제 유형을 사용하여 산술 연산을 수행할 수 있는 작업 표시
   double a=7,b=200;
   Print("---- 산술 연산 전");
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- a를 b로 나눔 (7/200)
```

© 2000-2025, MetaQuotes Ltd.

```
   a=a/b;
//--- 이제 변수 b의 값 복원을 에뮬레이트 함
   b=7.0/a; // 즉 b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 로 예상되지만 다름
//--- 새로 계산된 b 값을 출력
   Print("----- 산술 연산 후");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- 엑스퍼트 저널로 출력
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 (see that b is no more equal to 20

//--- 아주 작은 값 epsilon=1E-013 생성
   double epsilon=1e-13;
   Print("---- 아주 작은 값 생성");
   Print("epsilon=",epsilon); // epsilon=1E-013 가져오기
//--- 이제 b에서 epsilon을 뺀 후 다시 엑스퍼트 저널에 값을 출력
   b=b-epsilon;
//--- 두 가지 방법 사용
   Print("---- b 변수에서 epsilon을 뺀 후");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- 엑스퍼트 저널로 출력
// Print(b)=199.9999999999999  (이제 epsilon을 뺀 b의 값은 200으로 반올림할 수 없음)
// Print(DoubleToString(b,16))=199.9999999999998578
//    (이제 epsilon을 뺀 b 값은 200으로 반올림할 수 없음)
  }
```

추가 참조

DoubleToString, StringFormat

© 2000-2025, MetaQuotes Ltd.

PrintFormat

미리 설정된 형식에 따라 엑스퍼트 어드바이저 로그에 심볼과 값 세트를 포맷 및 인쇄.

```
void  PrintFormat(
   string format_string,   // 서식 string
   ...                     // 단순 유형의 값
   );
```

매개변수

format_string

[in] 서식 문자열은 단순 심볼로 구성되며, 서식 문자열 뒤에 인수가 있으면 서식 지정도 포함됩니다.

...

[in] 쉼표로 구분된 단순 유형의 값. 서식 문자열을 포함하여 총 매개변수의 수는 64개를 초과할 수 없습니다.

값 반환

String.

주의

최적화 하는 동안에는 전략 테스터에서 PrintFormat() 기능이 작동하지 않습니다.

매개변수의 수, 순서 및 유형은 수식어 집합과 정확히 일치해야하며, 그렇지 않으면 인쇄 결과가 정의되지 않습니다. PrintFormat() 대신 printf()를 사용할 수 있습니다.

서식 문자열 뒤에 매개변수가 있으면, 이 문자열에는 이러한 매개변수의 출력 서식을 나타내는 서식 규격이 포함되어야 합니다. 서식 지정은 항상 백분율 기호(%)로 시작합니다.

서식 문자열이 왼쪽에서 오른쪽으로 읽힙니다. 첫 번째 서식 지정이 충족되된(있는 경우), 서식 문자 열 이후의 첫 번째 매개변수 값이 사전 설정된 사양에 따라 변환되고 출력됩니다. 두 번째 형식 지정은 서식 문자열이 끝날 때까지 두 번째 매개변수의 변환 및 출력을 호출합니다.

서식 지정에는 다음과 같은 형식이 있습니다:

%[flags][width][.precision][{h | l | ll | I32 | I64}]type

서식 지정의 각 필드는 단순 심볼이거나 단순 서식 옵션을 나타내는 숫자입니다. 가장 간단한 서식 지정에는 백분율 기호(%)와 출력 매개변수의 유형 (예: %s)을 정의하는 기호만 포함됩니다. 서식 문자열에서 백분율 기호를 출력해야하는 경우 %%서식 사양을 사용하십시오.

플래그

플래그	설명	기본 동작
- (마이너스)	설정 폭 내 왼쪽 맞춤	오른쪽 맞춤
+ (플러스)	기호 유형 값에 대한 + 또는 -부호의 출력	값이 음수인 경우에만 부하가 표시됩니다

© 2000-2025, MetaQuotes Ltd.

플래그	설명	기본 동작
0 (영)	0이 사전 설정된 폭내의 출력 값 앞에 추가됩니다. 만약 **0** 플래그를 정수 형식 (**i, u, x, X, o, d**)으로 지정하고 정확도 사양(예: %04.d)을 설정한 경우 0은 무시됩니다.	아무것도 추가되지 않음
공백	기호 및 양수 값일 경우 출력 값 앞에 공백이 표시됩니다.	공백이 삽입되지 않았습니다
#	**o, x** 또는 **X** 서식과 함께 사용할 경우, 출력 값 0, 0x 또는 0X가 각각 추가되기 전에 발생합니다.	아무것도 추가되지 않음
	e, E, a 또는 **A** 서식과 함께 사용할 경우, 값은 항상 소수점으로 표시됩니다.	소수점은 0이 아닌 분수 부분이 있는 경우에만 표시됩니다.
	g 또는 **G** 서식과 함께 사용할 경우, 플래그는 출력 값에서 소수점의 존재를 정의하고 선행 0의 차단을 방지합니다. 플래그 #를 **c, d, i, u, s** 형식과 함께 사용하면 무시됩니다.	소수점은 0이 아닌 분수 부분이 있는 경우에만 표시됩니다. 선행 0이 차단되었습니다.

너비

서식화된 값의 최소 출력 심볼 수를 설정하는 음수가 아닌 소수입니다. 출력 기호 수가 지정된 폭보다 작으면 선형(flag -)에 따라 왼쪽 또는 오른쪽에서 해당 공백 수가 추가됩니다. 0 플래그가 있는 경우 출력 값 앞에 해당하는 0이 추가됩니다. 출력 심볼 수가 지정된 너비보다 크면 출력 값이 차단되지 않습니다.

별표(*)가 너비로 지정된 경우 전달된 매개변수 목록의 해당 위치에 int 유형 값을 표시해야 합니다. 출력 값의 폭을 지정하는 데 사용됩니다.

정밀도

출력 정밀도를 설정하는 음이 아닌 십진수 - 소수점 뒤의 자릿수입니다. 너비 사양과 구별되는 정밀도 사양은 반올림을 사용하거나 사용하지 않고 부분 유형의 부품을 절단할 수 있습니다.

정밀도 사양은 사용하는 서식 유형마다 다릅니다.

유형	설명	기본 동작
a, A	정밀도 규격은 소수점 뒤의 자릿수를 설정합니다.	기본 정밀도 - 6.
c, C	사용하지 않음	
d, i, u, o, x, X	최소 출력 자릿수를 설정. 해당 매개변수가 자릿수가 이 정밀도보다 작으면 출력값 왼쪽에 0이 추가됩니다. 출력 자릿수가 지정된 정밀도보다 크면 출력 값이 차단되지 않습니다.	기본 정밀도 - 1.
e, E, f	출력 자릿수를 소수점 뒤의 숫잘 설정합니다. 마지막 자릿수가 반올림됩니다.	기본 정밀도 - 6. 설정 정밀도가 0이거나 소수점이 없는 경우 소

© 2000-2025, MetaQuotes Ltd.

유형	설명	기본 동작
		수정이 표시되지 않습니다.
g, G	의미 있는 최대 수를 설정합니다.	6개의 의미 있는 숫자가 출력됩니다.
s	문자열의 출력 심볼 수를 설정. 문자열 길이가 정밀도를 초과하면 문자열이 잘립니다.	전체 문자열이 출력됩니다.

```
PrintFormat("1. %s", _Symbol);
PrintFormat("2. %.3s", _Symbol);
int length=4;
PrintFormat("3. %.*s", length, _Symbol);
/*
1. EURUSD
2. EUR
3. EURU
/
```

h | l | ll | I32 | I64

매개변수로 전달된 데이터 크기의 규격입니다.

매개변수 유형	사용된 접두사	유형의 공동 지정자
int	l (소문자 L)	d, i, o, x, 또는 X
uint	l (소문자 L)	o, u, x, 또는 X
long	ll (소문자 L 1개)	d, i, o, x, 또는 X
short	h	d, i, o, x, 또는 X
ushort	h	o, u, x, 또는 X
int	I32	d, i, o, x, 또는 X
uint	I32	o, u, x, 또는 X
long	I64	d, i, o, x, 또는 X
ulong	I64	o, u, x, 또는 X

type

서식 지정자는 서식화된 출력의 유일한 필수 필드입니다.

© 2000-2025, MetaQuotes Ltd.

심볼	유형	출력 서식
c	int	단축형 심볼 (유니코드)
C	int	char 유형의 심볼 (ANSI)
d	int	부호 있는 10진법 정수
i	int	부호 있는 10진법 정수
o	int	부호 없는 8진법 정수
u	int	부호 없는 10진법 정수
x	int	"abcdef"를 사용하는 부호없는 16진법 정수
X	int	"ABCDEF"를 사용하는 부호없는 16진법 정수
e	double	[-] d.ddde[sign] ddd 서식의 실제 값, 여기서 d - 소수 한 자리수, dddd - 소수 자릿수 한자리 이상, ddd - 지수, 플러스 또는 마이너스 기호의 크기를 결정하는 세자리 숫자
E	double	지수 부호가 대문자 (e 대신 **E**)로 출력된다는 점을 제외하고 e의 형식과 유사함
f	double	[-] dddd.dddd 서식의 실제 값, 여기서 dddd - 하나 이상의 소수 자릿수. 소수점 이전의 표시된 자릿수는 숫자 값의 크기에 따라 다릅니다. 소수점 뒤의 자릿수는 필요한 정확도에 따라 달라집니다.
g	double	더 압축된 출력에 따라 **f** 또는 **e** 서식의 실제 값 출력.
G	double	더 압축된 출력에 따라 **F** 또는 **E** 서식의 실제 값 출력.
a	double	[-]0xh.hhhh **p**±dd 서식의 실수, 여기서 h.hhhh 는 "abcdef"를 사용하는 16진법 형태의 가수 , dd는 하나 이상의 지수. 소수 자릿수는 정확도 사양에 따라 결정
A	double	[-]0xh.hhhh **P**±dd 서식의 실수, 여기서 h.hhhh는 "ABCDEF"를 사용하는 16진법 형태의 가수, dd는 하나 이상의 지수. 소수 자릿수는 정확도 사양에 따라 결정
s	string	문자열 출력

PrintFormat() 대신 printf()를 사용할 수 있습니다.

예를 들어:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- trade server name
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- account number
   int login=(int)AccountInfoInteger(ACCOUNT_LOGIN);
//--- long value output
   long leverage=AccountInfoInteger(ACCOUNT_LEVERAGE);
   PrintFormat("%s %d: leverage = 1:%I64d",
               server,login,leverage);
//--- account currency
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- double value output with 2 digits after the decimal point
   double equity=AccountInfoDouble(ACCOUNT_EQUITY);
   PrintFormat("%s %d: account equity = %.2f %s",
               server,login,equity,currency);
//--- double value output with mandatory output of the +/- sign
   double profit=AccountInfoDouble(ACCOUNT_PROFIT);
   PrintFormat("%s %d: current result for open positions = %+.2f %s",
               server,login,profit,currency);
//--- double value output with variable number of digits after the decimal point
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   string format_string=StringFormat("%%s: point value  = %%.%df",_Digits);
   PrintFormat(format_string,_Symbol,point_value);
//--- int value output
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   PrintFormat("%s: current spread in points = %d ",
               _Symbol,spread);
//--- double value output in the scientific (floating point) format with 17 meaningful
   PrintFormat("DBL_MAX = %.17e",DBL_MAX);
//--- double value output in the scientific (floating point) format with 17 meaningful
   PrintFormat("EMPTY_VALUE = %.17e",EMPTY_VALUE);
//--- output using PrintFormat() with default accuracy
   PrintFormat("PrintFormat(EMPTY_VALUE) = %e",EMPTY_VALUE);
//--- simple output using Print()
   Print("Print(EMPTY_VALUE) = ",EMPTY_VALUE);
/* execution result
   MetaQuotes-Demo 1889998: leverage = 1:100
   MetaQuotes-Demo 1889998: account equity = 22139.86 USD
   MetaQuotes-Demo 1889998: current result for open positions = +174.00 USD
   EURUSD: point value  = 0.00001
   EURUSD: current spread in points = 12
   DBL_MAX = 1.79769313486231570e+308
   EMPTY_VALUE = 1.79769313486231570e+308
   PrintFormat(EMPTY_VALUE) = 1.797693e+308
   Print(EMPTY_VALUE) = 1.797693134862316e+308
*/
  }
```

추가 참조

StringFormat, DoubleToString, Real types (double, float)

© 2000-2025, MetaQuotes Ltd.

ResetLastError

사전 정의된 변수 _LastError의 값을 0으로 설정.

```
void  ResetLastError();
```

값 반환

반환 값 없음.

주의

GetLastError() 함수는 _LastError 변수를 0으로 설정하지 않습니다. 일반적으로 함수를 호출하기 전에 ResetLastError() 기능을 호출하고, 그 후 오류 형태를 확인합니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 함수를 호출하기 전에 마지막 오류 코드를 재설정합니다.
//--- 그렇지 않으면 GetLastError()가 이전 오류 코드를 반환할 수 있습니다.
   long lres=SymbolInfoInteger("123456",SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
   lres=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
   ResetLastError();
   lres=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   PrintFormat("lres=%d  error=%d",lres,GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

ResourceCreate

데이터 세트를 기반으로 이미지 리소스를 생성. 이 기능에는 두 가지 변수가 있습니다:
파일을 기반으로 리소스 생성

```
bool  ResourceCreate(
   const string      resource_name,       // 리소스 이름
   const string      path                 // 파일의 상대 경로
   );
```

픽셀 배열을 기반으로 리소스 생성

```
bool  ResourceCreate(
   const string      resource_name,       // 리소스 이름
   const uint&       data[],              // 데이터 집합 배열
   uint              img_width,           // 이미지 리소스의 폭
   uint              img_height,          // 이미지 리소스 높이
   uint              data_xoffset,        // 이미지의 왼쪽 상단 모서리의 수평 오른쪽 오프셋
   uint              data_yoffset,        // 이미지 왼쪽 상단 모서리의 수직 하향 오프셋
   uint              data_width,          // 데이터 세트를 기준으로 한 이미지 총 너비
   ENUM_COLOR_FORMAT color_format         // 색 처리 방법
   );
```

매개변수

resource_name

[in] 리소스 이름.

data[][]

[in] 전체 이미지를 만들기 위한 1차원 또는 2차원 배열..

img_width

[in] 이미지 형식으로 리소스에 배치될 직사각형 이미지 영역의 너비(픽셀). *data_width 값보다 클 수 없습니다.*

img_height

[in] 이미지 형식으로 리소스에 배치할 직사각형 이미지 영역의 높이(픽셀).

data_xoffset

[in] 이미지의 직사각형 영역의 오른쪽 수평 간격 띄우기.

data_yoffset

[in] 이미지의 직사각형 영역의 수직 아랫쪽 간격 띄우기.

data_width

[in] 1차원 배열에만 필요. 데이터 세트에서 이미지 전체 너비를 나타냅니다. *data_width*=0 이면, *img_width와 같은 것으로 간주됩니다.* 2차원 배열의 경우 매개변수는 무시되며 *data[]* 배열의 두 번째 차원과 동일하다고 가정합니다.

color_format

[in] ENUM_COLOR_FORMAT 열거의 값에서의 색 처리 방법.

© 2000-2025, MetaQuotes Ltd.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출. 다음과 같은 오류가 발생할 수 있습니다:

- 4015 – ERR_RESOURCE_NAME_DUPLICATED (동적 및 정적 리소스의 동일한 이름)
- 4016 – ERR_RESOURCE_NOT_FOUND (리소스를 찾을 수 없음)
- 4017 – ERR_RESOURCE_UNSUPPORTED_TYPE (이 리소스 유형이 지원되지 않음)
- 4018 – ERR_RESOURCE_NAME_IS_TOO_LONG (리소스 이름이 너무 긺)

주의

두 번째 버전의 함수가 서로 다른 너비, 높이 및 이동 매개변수를 사용하여 동일한 리소스를 생성하도록 요청하면 새 리소스가 생성되지 않고 기존 리소스를 업데이트하기만 하면 됩니다.

기능의 첫 번째 버전은 파일로부터 이미지와 소리를 업로드하는 데 사용되며, 두 번째 버전은 이미지 동적 생성에만 사용됩니다.

이미지는 24비트 또는 32비트의 색 심도를 가진 BMP 형식이어야 합니다. 사운드는 WAV 형식에서만 사용할 수 있습니다. 리소스의 크기는 16 Mb를 초과할 수 없습니다.

ENUM_COLOR_FORMAT

식별자	설명
COLOR_FORMAT_XRGB_NOALPHA	알파 채널의 구성 요소는 무시됩니다.
COLOR_FORMAT_ARGB_RAW	색상 구성 요소가 단자에서 처리되지 않습니다 (사용자가 올바르게 설정해야함)
COLOR_FORMAT_ARGB_NORMALIZE	색상 구성요소는 단말기에서 처리됩니다

예:

```
//+------------------------------------------------------------------+
//| 그래픽 리소스 데이터 업데이트                                    |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- 0차원이 전달되면 종료
   if(width==0 || height==0)
      return;
//--- 리소스 데이터를 업데이트하고 차트를 다시 그립니다.
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALI
      ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 그래픽 리소스의 매개변수 선언
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- 픽셀 배열의 크기를 설정합니다.
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- 픽셀 배열을 투명한 색상으로 채우고 이를 기반으로 그래픽 리소스를 만듭니다.
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- 생성된 그래픽 리소스를 확인합니다.
//--- 현재 바의 시간과 가격 데이터를 가져옵니다.
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- 마지막 틱 가격과 시간의 좌표를 사용하여 Bitmap 객체를 생성합니다.
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 생성된 비트맵 객체의 너비와 높이를 그래픽 리소스의 너비와 높이와 동일하게 설정합니다.
//--- 객체 앵커 포인트를 중심으로 설정합니다.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- 비트맵 객체에 대해 이전에 생성된 그래픽 리소스를 이미지 파일로 지정합니다.
//--- 이 경우 사용된 그래픽 리소스의 이름을 나타내기 위해 이름 앞에 "::"를 추가해야 합니다.
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- 투명도 200으로 DodgerBlue 색상을 설정합니다.
   uint clr=ColorToARGB(clrDodgerBlue,200);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 그래픽 리소스의 전체 픽셀 배열을 설정된 색상으로 채웁니다.
   ArrayInitialize(rc_data,clr);
//--- 그래픽 리소스 데이터 업데이트
   Update(rc_name,rc_data,rc_width,rc_height,true);
  }
```

추가 참조

Resources, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceFree

함수는 동적으로 생성된 리소스를 삭제 (해당 리소스에 할당된 메모리를 확보).

```
bool  ResourceFree(
   const string  resource_name      // 리소스 이름
   );
```

매개변수

resource_name

[in] 리소스 이름은 "::"로 시작합니다.

값 반환

성공하면 true, 실패하면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

ResourceFree()를 사용하면 mql5 애플리케이션 개발자가 리소스를 적극적으로 사용할 때 메모리 사용을 관리할 수 있습니다. 그래픽 개체는 메모리에서 삭제할 리소스에 바인딩 되며, 삭제된 후 올바르게 표시됩니다. 그러나 새로 생성된 그래픽 개체 (OBJ_BITMAP 및 OBJ_BITMAP_LABEL) 는 삭제된 리소스를 사용할 수 없습니다.

이 함수는 프로그램에서 생성한 동적 리소스만 삭제합니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 리소스의 매개변수 선언
   string rc_name="Resource";
   uint   rc_width=100;
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- 픽셀 배열의 크기를 설정합니다.
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- 픽셀 배열을 투명한 색상으로 채우고 이를 기반으로 그래픽 리소스를 만듭니다.
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- 생성된 그래픽 리소스를 확인합니다.
//--- 현재 바의 시간과 가격 데이터를 가져옵니다.
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- 마지막 틱 가격과 시간의 좌표를 사용하여 Bitmap 객체를 생성합니다.
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 생성된 비트맵 객체의 너비와 높이를 그래픽 리소스의 너비와 높이와 동일하게 설정합니다.
//--- 객체 앵커 포인트를 중심으로 설정합니다.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- 비트맵 객체에 대해 이전에 생성된 그래픽 리소스를 이미지 파일로 지정합니다.
//--- 이 경우 사용된 그래픽 리소스의 이름을 나타내기 위해 이름 앞에 "::"를 추가해야 합니다.
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- 투명도 200으로 DodgerBlue 색상을 설정합니다.
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- 그래픽 리소스의 전체 픽셀 배열을 설정된 색상으로 채웁니다.
   ArrayInitialize(rc_data,clr);
//--- 그래픽 리소스 데이터 업데이트
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- 3초 정도 기다린 후 이미지 색상을 변경합니다.
   Print("Wait 3 seconds before changing color");
   Sleep(3000);
//--- 투명도 200으로 OrangeRed 색상을 설정합니다.
   Print("Change color");
   clr=ColorToARGB(clrOrangeRed,200);
//--- 그래픽 리소스의 전체 픽셀 배열을 설정된 색상으로 채웁니다.
   ArrayInitialize(rc_data,clr);
//--- 그래픽 리소스 데이터 업데이트
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- 3초간 기다린 후 그래픽 리소스를 해제합니다.
   Print("Wait 3 seconds before ResourceFree()");
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(3000);
   bool res=ResourceFree("::"+rc_name);
   Print("ResourceFree: ",res);

//--- 리소스를 해제한 후 색상을 변경해 보세요.
   Print("Trying to change color to GreenYellow after ResourceFree()");
//--- 투명도 200으로 GreenYellow 색상을 설정합니다.
   clr=ColorToARGB(clrGreenYellow,200);
//--- 그래픽 리소스의 전체 픽셀 배열을 설정된 색상으로 채웁니다.
   ArrayInitialize(rc_data,clr);
//--- 그래픽 리소스 데이터 업데이트(이미지는 유지되지만 색상은 변경할 수 없음)
   Update(rc_name,rc_data,rc_width,rc_height,true);
   Print("The color has not changed because the resource has been released");

//--- 3초 정도 기다린 후 비트맵 객체를 삭제합니다.
   Print("Wait 3 seconds before deleting the Bitmap object");
   Sleep(3000);
   Print("Delete Bitmap object");
   ObjectDelete(0,obj_name);
  }
//+------------------------------------------------------------------+
//| 그래픽 리소스 데이터 업데이트                                    |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- 0차원이 전달되면 종료
   if(width==0 || height==0)
      return;
//--- 리소스 데이터를 업데이트하고 차트를 다시 그립니다.
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALI
      ChartRedraw();
  }
```

추가 참조

Resources, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceReadImage

이 함수는 ResourceCreate() 함수에 의해 생성 되거나 컴파일 중에 EX5 파일에 저장된 그래픽 리소스에서 데이터를 읽습니다.

```
bool  ResourceReadImage(
   const string      resource_name,       // 읽을 수 있는 그래픽 리소스 이름
   uint&             data[],              // 리소스에서 데이터를 수신하기 위한 배열
   uint&             width,               // 리소스에서 이미지 너비를 수신
   uint&             height,              // 리소스에서 이미지 높이를 수신
   );
```

매개변수

resource_name

[in] 이미지를 포함하는 그래픽 리소스의 이름. 자체 리소스에 액세스하기 위해 이름은 "::resourcename" 형식으로 사용됩니다. 컴파일된 EX5 파일에서 리소스를 다운로드하는 경우 MQL5 디렉토리, 파일 및 리소스 이름에 대한 경로("path₩₩filename.ex5::resourcename")와 함께 전체 이름을 사용해야 합니다.

data[][]

[in] 그래픽 리소스에서 데이터를 수신하기 위한 1차원 또는 2차원 배열.

img_width

[out] 그래픽 리소스 이미지 너비(픽셀).

img_height

[out] 그래픽 리소스 이미지 높이(픽셀).

값 반환

성공하면 true, 그렇지 않은 경우 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

data[]를 그래픽 리소스를 생성하는데 사용하여, COLOR_FORMAT_ARGB_NORMALIZE 또는 COLOR_FORMAT_XRGB_NOALPHA 색 형식을 사용해야 합니다.

data[] 배열이 2차원이고 두 번째 차원이 X(폭) 그래픽 리소스 크기보다 작으면 ResourceReadImage() 함수가 false를 반환하고 읽기는 수행되지 않습니다. 그러나 리소스가 있으면 실제 이미지 크기가 폭 및 높이 매개변수로 돌아갑니다. 이를 통해 리소스에서 데이터를 다시 수신할 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 리소스의 매개변수 선언
   string rc_name="Resource";
   uint   rc_width=100;
```

© 2000-2025, MetaQuotes Ltd.

```
   uint   rc_height=100;
   uint   rc_data[];
   uint   rc_size=rc_width*rc_height;

   ResetLastError();
//--- 픽셀 배열의 크기를 설정합니다.
   if(ArrayResize(rc_data,rc_size)!=rc_size)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- 픽셀 배열을 투명한 색상으로 채우고 이를 기반으로 그래픽 리소스를 만듭니다.
   ArrayInitialize(rc_data,0x00FFFFFF);
   if(!ResourceCreate(rc_name,rc_data,rc_width,rc_height,0,0,0,COLOR_FORMAT_ARGB_NORMA
     {
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",rc_data.Size());

//--- 생성된 그래픽 리소스를 확인합니다.
//--- 현재 바의 시간과 가격 데이터를 가져옵니다.
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- 마지막 틱 가격과 시간의 좌표를 사용하여 Bitmap 객체를 생성합니다.
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 생성된 비트맵 객체의 너비와 높이를 그래픽 리소스의 너비와 높이와 동일하게 설정합니다.
//--- 객체 앵커 포인트를 중심으로 설정합니다.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- 비트맵 객체에 대해 이전에 생성된 그래픽 리소스를 이미지 파일로 지정합니다.
//--- 이 경우 사용된 그래픽 리소스의 이름을 나타내기 위해 이름 앞에 "::"를 추가해야 합니다.
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+rc_name);

//--- 투명도 200으로 DodgerBlue 색상을 설정합니다.
   uint clr=ColorToARGB(clrDodgerBlue,200);
//--- 그래픽 리소스의 전체 픽셀 배열을 설정된 색상으로 채웁니다.
   ArrayInitialize(rc_data,clr);
//--- 그래픽 리소스 데이터 업데이트
```

© 2000-2025, MetaQuotes Ltd.

```
   Update(rc_name,rc_data,rc_width,rc_height,true);

//--- 그래픽 리소스 데이터를 읽기 전에 3초 동안 기다립니다.
   Print("Wait 3 seconds before ResourceReadImage()");
   Sleep(3000);
//--- 리소스의 이미지를 새로운 픽셀 배열로 읽습니다.
   uint rc_data_copy[];
   uint w=0,h=0;
   ResetLastError();
   if(!ResourceReadImage("::"+rc_name,rc_data_copy,w,h))
     {
      Print("ResourceReadImage() failed. Error code: ",GetLastError());
      return;
     }

//--- 투명도 200으로 OrangeRed 색상을 설정합니다.
   clr=ColorToARGB(clrOrangeRed,200);
//--- 그래픽 리소스의 전체 픽셀 배열을 설정된 색상으로 채우고 이를 기반으로 새로운 그래픽 리소스를
   ArrayInitialize(rc_data_copy,clr);
   if(!ResourceCreate(rc_name+"Copy",rc_data_copy,rc_width,rc_height,0,0,0,COLOR_FORMA
     {
      Print("New ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created new recource array: ",rc_data_copy.Size());

//--- 마지막 틱 가격과 시간의 좌표를 사용하여 "그래픽 라벨" 객체를 생성합니다.
   string obj_name2="BitmapLabel";
   if(!ObjectCreate(0,obj_name2,OBJ_BITMAP_LABEL,0,0,0))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 이전에 받은 가격과 시간을 이용하여 화면 좌표를 가져옵니다.
   int x=0,y=0;
   if(!ChartTimePriceToXY(0,0,tick.time,tick.bid,x,y))
     {
      Print("New ChartTimePriceToXY() failed. Error code: ",GetLastError());
      return;
     }
//--- 생성된 그래픽 라벨 객체의 너비와 높이를 그래픽 리소스의 너비와 높이와 동일하게 설정합니다.
//--- 객체 앵커 포인트를 중심으로 설정합니다.
   ObjectSetInteger(0,obj_name2,OBJPROP_XSIZE,rc_width);
   ObjectSetInteger(0,obj_name2,OBJPROP_YSIZE,rc_height);
   ObjectSetInteger(0,obj_name2,OBJPROP_ANCHOR,ANCHOR_LEFT_UPPER);
   ObjectSetInteger(0,obj_name2,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(0,obj_name2,OBJPROP_YDISTANCE,y);
//--- 복사된 그래픽 리소스를 그래픽 라벨 객체의 이미지 파일로 설정합니다.
//--- 이 경우 사용된 그래픽 리소스의 이름을 나타내기 위해 이름 앞에 "::"를 추가해야 합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetString(0,obj_name2,OBJPROP_BMPFILE,"::"+rc_name+"Copy");

//--- 새 그래픽 라벨 객체의 색상을 변경합니다.
   Print("Wait 3 seconds before changing color to GreenYellow");
   Sleep(3000);
//--- 투명도 200으로 GreenYellow 색상을 설정합니다.
   clr=ColorToARGB(clrGreenYellow,200);
//--- 새 그래픽 리소스의 전체 픽셀 배열을 설정된 색상으로 채웁니다.
   ArrayInitialize(rc_data_copy,clr);
//--- 그래픽 리소스 데이터 업데이트
   Update(rc_name+"Copy",rc_data_copy,rc_width,rc_height,true);

//--- 3초 동안 기다린 후 리소스와 두 객체를 모두 삭제합니다.
   Print("Wait 3 seconds before deleting both objects");
   Sleep(3000);
   Print("Deleting Resource and all Bitmap objects");
   ResourceFree("::"+rc_name);
   ResourceFree("::"+rc_name+"Copy");
   ObjectDelete(0,obj_name);
   ObjectDelete(0,obj_name2);
  }
//+------------------------------------------------------------------+
//| 그래픽 리소스 데이터 업데이트                                    |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
  {
//--- 0차원이 전달되면 종료
   if(width==0 || height==0)
      return;
//--- 리소스 데이터를 업데이트하고 차트를 다시 그립니다.
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALI
      ChartRedraw();
  }
```

추가 참조

Resource, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

ResourceSave

지정된 파일에 리소스를 저장.

```
bool  ResourceSave(
   const string  resource_name      // 리소스 이름
   const string  file_name          // 파일 이름
   );
```

매개변수

resource_name

[in] 리소스 이름은 "::"로 시작해야 합니다.

file_name

[in] MQL5₩Files에 상대적인 파일의 이름.

값 반환

성공하는 경우 true, 그렇지 않으면 false. 오류 정보는 GetLastError()로 문의.

주의

이 함수는 항상 파일을 덮어쓰고 필요한 경우 파일 이름에 필요한 모든 중간 디렉터리를 만듭니다.

예:

```
//--- 그래픽 리소스 매개변수
string ExtResName="Resource";
int    ExtResWidth=100;
int    ExtResHeight=100;
uint   ExtResData[];
int    ExtResSize=ExtResWidth*ExtResHeight;

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   ResetLastError();
//--- 픽셀 배열의 크기를 설정합니다.
   if(ArrayResize(ExtResData,ExtResSize)!=ExtResSize)
     {
      Print("ArrayResize() failed. Error code: ",GetLastError());
      return;
     }
//--- 픽셀 배열을 투명한 색상으로 채우고 이를 기반으로 그래픽 리소스를 만듭니다.
   ArrayInitialize(ExtResData,0x00FFFFFF);
   if(!ResourceCreate(ExtResName,ExtResData,ExtResWidth,ExtResHeight,0,0,0,COLOR_FORMA
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("ResourceCreate() failed. Error code: ",GetLastError());
      return;
     }
   Print("Size of created recource array: ",ExtResData.Size());

//--- 생성된 그래픽 리소스를 확인합니다.
//--- 현재 바의 시간과 가격 데이터를 가져옵니다.
   MqlTick tick={};
   if(!SymbolInfoTick(Symbol(),tick))
     {
      Print("SymbolInfoTick() failed. Error code: ",GetLastError());
      return;
     }
//--- 마지막 틱 가격과 시간의 좌표를 사용하여 Bitmap 객체를 생성합니다.
   string obj_name="Bitmap";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP,0,tick.time,tick.bid))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 생성된 비트맵 객체의 너비와 높이를 그래픽 리소스의 너비와 높이와 동일하게 설정합니다.
//--- 객체 앵커 포인트를 중심으로 설정합니다.
   ObjectSetInteger(0,obj_name,OBJPROP_XSIZE,ExtResWidth);
   ObjectSetInteger(0,obj_name,OBJPROP_YSIZE,ExtResHeight);
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- 비트맵 객체에 대해 이전에 생성된 그래픽 리소스를 이미지 파일로 지정합니다.
//--- 이 경우 사용된 그래픽 리소스의 이름을 나타내기 위해 이름 앞에 "::"를 추가해야 합니다.
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"::"+ExtResName);

//--- 투명도 200으로 GreenYellow 색상을 설정합니다.
   uint clr=ColorToARGB(clrGreenYellow,200);
//--- 그래픽 리소스의 전체 픽셀 배열을 설정된 색상으로 채웁니다.
   ArrayInitialize(ExtResData,clr);
//--- DodgerBlue 색상을 사용하여 그리드 그리기
   for(int x=0;x<ExtResWidth;x+=9)
      LineVertical(x,0,ExtResHeight,ColorToARGB(clrDodgerBlue,200));
   for(int y=0;y<ExtResHeight;y+=9)
      LineHorizontal(0,ExtResWidth,y,ColorToARGB(clrDodgerBlue,200));
//--- 그래픽 리소스 데이터 업데이트
   Update(ExtResName,ExtResData,ExtResWidth,ExtResHeight,true);

//--- 리소스 및 그래픽 객체를 삭제하기 전에 3초 동안 기다립니다.
   Print("Wait 3 seconds before deleting the resource and graphic object");
   Sleep(3000);

//--- 리소스를 파일에 저장
   ResetLastError();
   if(!ResourceSave("::"+ExtResName,"ResourceSave\\"+ExtResName+".bmp"))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("ResourceSave() failed. Error code: ",GetLastError());
      return;
     }
//--- 리소스 및 그래픽 객체 이미지 삭제
   if(!ResourceFree("::"+ExtResName))
     {
      Print("ResourceFree() failed. Error code: ",GetLastError());
      return;
     }
   if(!ObjectDelete(0,obj_name))
     {
      Print("ObjectDelete() failed. Error code: ",GetLastError());
      return;
     }

//--- 이제 이미지 너비만큼 왼쪽으로 이동한 마지막 틱의 가격과 시간 좌표를 사용하여 "그래픽 라벨"
//--- 이전에 받은 마지막 틱 가격과 시간을 사용하여 화면 좌표를 가져옵니다.
   int x=0,y=0;
   if(!ChartTimePriceToXY(0,0,tick.time,tick.bid,x,y))
     {
      Print("ChartTimePriceToXY() failed. Error code: ",GetLastError());
      return;
     }
   obj_name="BitmapLabel";
   if(!ObjectCreate(0,obj_name,OBJ_BITMAP_LABEL,0,0,0))
     {
      Print("ObjectCreate() failed. Error code: ",GetLastError());
      return;
     }
//--- 객체 앵커 포인트를 중심으로 설정합니다.
   ObjectSetInteger(0,obj_name,OBJPROP_ANCHOR,ANCHOR_CENTER);
//--- 이미지 너비만큼 왼쪽으로 이동한 마지막 틱의 가격 및 시간 좌표를 그래픽 라벨 객체에 설정합니
   ObjectSetInteger(0,obj_name,OBJPROP_XDISTANCE,x-ExtResWidth);
   ObjectSetInteger(0,obj_name,OBJPROP_YDISTANCE,y);
//--- 이전에 bmp 이미지로 저장한 리소스를 그래픽 라벨 객체에 대해 이미지 파일로 지정합니다.
   ObjectSetString(0,obj_name,OBJPROP_BMPFILE,"\\Files\\ResourceSave\\"+ExtResName+".b
//--- 차트 업데이트
   ChartRedraw();

//--- 그래픽 라벨 객체를 삭제하기 전에 3초 동안 기다립니다.
   Print("Wait 3 seconds before deleting the new graphic object");
   Sleep(3000);
   if(!ObjectDelete(0,obj_name))
      Print("ObjectDelete() failed. Error code: ",GetLastError());
  }
//+------------------------------------------------------------------+
//| 그래픽 리소스 데이터 업데이트                                    |
//+------------------------------------------------------------------+
void Update(const string res_name,const uint &pixel_data[],const uint width,const uint
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 0차원이 전달되면 종료
   if(width==0 || height==0)
      return;
//--- 리소스 데이터를 업데이트하고 차트를 다시 그립니다.
   if(ResourceCreate(res_name,pixel_data,width,height,0,0,0,COLOR_FORMAT_ARGB_NORMALI
      ChartRedraw();
  }
//+------------------------------------------------------------------+
//| 수직선을 그립니다                                                |
//+------------------------------------------------------------------+
void LineVertical(int x,int y1,int y2,const uint clr)
  {
   int tmp;
//--- Y로 정렬
   if(y1>y2)
     {
      tmp=y1;
      y1 =y2;
      y2 =tmp;
     }
//--- 이미지 경계 밖의 선
   if(y2<0 || y1>=ExtResHeight || x<0 || x>=ExtResWidth)
      return;
//--- 이미지 경계 내에 머무르세요
   if(y1<0)
      y1=0;
   if(y2>=ExtResHeight)
      y2=ExtResHeight-1;
//--- 선을 그립니다
   int index=y1*ExtResWidth+x;
   for(int i=y1; i<=y2; i++,index+=ExtResWidth)
      ExtResData[index]=clr;
  }
//+------------------------------------------------------------------+
//| 수평선 그리기                                                    |
//+------------------------------------------------------------------+
void LineHorizontal(int x1,int x2,int y,const uint clr)
  {
   int tmp;
//--- X로 정렬
   if(x1>x2)
     {
      tmp=x1;
      x1 =x2;
      x2 =tmp;
     }
//--- 이미지 경계 밖의 선
   if(x2<0 || x1>=ExtResWidth || y<0 || y>=ExtResHeight)
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
//--- 이미지 경계 내에 머무르세요
   if(x1<0)
      x1=0;
   if(x2>=ExtResWidth)
      x2=ExtResWidth-1;
//--- 선을 그립니다
   ArrayFill(ExtResData,y*ExtResWidth+x1,(x2-x1)+1,clr);
  }
```

추가 참조

Resources, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

© 2000-2025, MetaQuotes Ltd.

SetReturnError

작업을 완료할 때 터미널 프로세스를 반환하는 코드를 설정

```
void  SetReturnError(
   int ret_code      // 클라이언트 터미널 완료 코드
   );
```

매개변수

ret_code

[in] 작업을 완료할 때 클라이언트 터미널 프로세스에서 반환할 코드.

값 반환

반환 값 없음.

주의

지정된 ***ret_code*** 반환 코드를 설정하는데 SetReturnError() 기능을 사용하면 명령줄을 통해 터미널을 시작할 때 프로그래밍 작업이 완료된 이유를 분석할 때 유용합니다.

TerminalClose()와 달리, SetReturnError() 기능은 터미널 작업을 완료하지 않습니다. 대신, 터미널 프로세스가 완료되면 반환하는 코드만 설정합니다.

SetReturnError() 기능을 여러 번 호출하거나 다른 MQL5 프로그램에서 호출하는 경우 터미널은 마지막 세트 반환 코드를 반환합니다.

설정된 코드는 다음의 경우를 제외하고 터미널 프로세스 완료 시 반환됩니다:

- 실행 중심각한 오류가 발생;
- TerminalClose(int ret_code) 기능은 지정된 코드로 터미널 작동 완료 명령을 실행하는 기능이 호출됨.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   matrix matrix_a =
     {
        {-3.474589, 1.106384, -9.091977,-3.925227 },
        {-5.522139, 2.366887,-15.162351,-6.357512 },
        { 8.394926,-2.960067, 22.292115, 9.524129 },
        { 7.803242,-2.080287, 19.217706, 8.186645 }
     };
   matrix matrix_l(4,4);
   matrix matrix_u(4,4);

//--- LU 분해
   matrix_a.LU(matrix_l,matrix_u);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- A = L * U인지 체크
   matrix matrix_lu=matrix_l.MatMul(matrix_u);
   int    compare_errors=(int)matrix_a.Compare(matrix_lu,1e-29);
   Print("MatrixCompare errors=",compare_errors);

//--- 완료되면 클라이언트 터미널은 두 행렬을 비교하는 동안 발생한 오류 수를 반환합니다.
   SetReturnError(compare_errors);
  }
```

추가 참조

Program Running, Runtime Errors, Uninitialization Reason Codes, TerminalClose

© 2000-2025, MetaQuotes Ltd.

SetUserError

사전 정의된 변수 _LastError를 ERR_USER_ERROR_FIRST + user_error와 동일한 값을 설정합니다.

```
void  SetUserError(
   ushort user_error,   // 오류 번호
   );
```

매개변수

user_error

[in] 사용자가 설정한 오류 번호.

값 반환

반환 값 없음.

주의

SetUserError(user_error) 기능을 사용하여 오류를 설정한 후, GetLastError()는 ERR_USER_ERROR_FIRST + user_error.와 동일한 값을 반환합니다.

예를 들어:

```
void OnStart()
  {
//--- 오류 번호 설정 65537=(ERR_USER_ERROR_FIRST +1)
   SetUserError(1);
//--- 마지막 오류 코드 가져오기
   Print("GetLastError = ",GetLastError());
/*
  결과
   GetLastError = 65537
*/
  }
```

© 2000-2025, MetaQuotes Ltd.

Sleep

이 기능은 지정된 간격 내에 현재 엑스퍼트 어드바이저 또는 스크립트의 실행을 일시 중단.

```
void  Sleep(
   int  milliseconds      // 간격
   );
```

매개변수

밀리초

[in] 지연 간격(밀리초).

값 반환

반환 값 없음.

주의

지표는 인터페이스 스레드에서 실행되며 속도가 느려서는 안되므로 사용자 지정 지표에 대한 Sleep() 기능을 호출할 수 없습니다. 이 기능에는 0.1초마다 EA 중지 플래그가 내장되어 있습니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 차트의 댓글에 10부터 1까지의 카운트다운을 표시합니다.
   for(int i=10; i>0 && !IsStopped(); i--)
     {
      Comment(StringFormat("Wait %u seconds",i));
      Sleep(1000);
     }
//--- 스크립트의 목적을 설명하는 "도착" 주석에 텍스트를 작성합니다.
   string text="This was a test showing how the Sleep() function works";
   string mess="";
   for(int i=0; i<(int)text.Length(); i++)
     {
      mess+=ShortToString(text.GetChar(i));
      Sleep(100);
      Comment(mess);
     }
//--- 안녕...
   Sleep(1000);
   for(int i=0; i<6; i++)
     {
      mess=(i % 2 == 0 ? "" : "  Bye!");
      Comment(mess);
```

© 2000-2025, MetaQuotes Ltd.

```
      Sleep(300);
     }
//--- 차트의 텍스트를 삭제합니다.
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalClose

이 기능은 터미널의 작동을 명령합니다.

```
bool  TerminalClose(
   int ret_code      // 클라이언트 터미널의 종료 코드
   );
```

매개변수

ret_code

[in] 반환 코드, 작업 완료 시 클라이언트 터미널의 프로세스에서 반환됩니다.

값 반환

이 함수는 성공하면 true를, 그렇지 않으면 false를 반환합니다.

주의

TerminalClose() 기능은 터미널을 즉시 정지시키는 것이 아니라 터미널의 작동을 명령할 뿐 입니다.

TerminalClose()라고 하는 엑스퍼트 어드바이저의 코드는 즉시 완료를 위한 모든 준비가 되어 있어야 합니다(예: 이전에 열었던 모든 파일은 일반 모드에서 닫아야 함). 이 기능의 호출은 반환 작업자가 직접 수행해야 합니다.

ret_code 매개변수는 명령 프롬프트에서 시작할 때 터미널 작업의 프로그램 종료 이유를 분석하는 데 필요한 반환 코드를 나타낼 수 있습니다.

예를 들어:

```
//--- 매개변수 입력
input int  tiks_before=500; // 종료 때까지의 틱 수
input int  pips_to_go=15;   // pi ps 거리
input int  seconds_st=50;   // 엑스퍼트 어드바이저에게 주어진 시간(초)
//--- 세계의
datetime   launch_time;
int        tick_counter=0;
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," reason code = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
   static double first_bid=0.0;
   MqlTick       tick;
```

© 2000-2025, MetaQuotes Ltd.

```
   double         distance;
//---
   SymbolInfoTick(_Symbol,tick);
   tick_counter++;
   if(first_bid==0.0)
     {
      launch_time=tick.time;
      first_bid=tick.bid;
      Print("first_bid =",first_bid);
      return;
     }
//--- pips 가격 거리
   distance=(tick.bid-first_bid)/_Point;
//--- EA 작동을 추적하기 위한 알림을 표시
   string comm="From the moment of start:\r\n\x25CF elapsed seconds: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF ticks received: "+(string)tick_counter+" ;"+
               "\r\n\x25CF price went in points: "+StringFormat("%G",distance);
   Comment(comm);
//--- 터미널 닫힘 상태 점검 섹션
   if(tick_counter>=tiks_before)
      TerminalClose(0);    // 틱 카운터로 나감
   if(distance>pips_to_go)
      TerminalClose(1);    // pips_to_go와 같은 pips의 수로 올라감
   if(distance<-pips_to_go)
      TerminalClose(-1);   // pips_to_go와 같은 pips 수만큼 내려감
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);  // 시간제한으로 종료
//---
  }
```

추가 참조

Program running, Execution errors, Reasons for deinitialization

© 2000-2025, MetaQuotes Ltd.

TesterHideIndicators

EA에서 사용되는 지표를 표시하거나 숨기는 모드를 설정. 이 기능은 시험 중에만 사용된 지표의 가시성을 관리하기 위한 것입니다.

```
void  TesterHideIndicators(
   bool      hide     // 플래그
   );
```

매개변수

숨김

[in] 테스트할 때 지표를 숨기기 위한 플래그. 생성된 지표를 숨기려면 true를, 그렇지 않으면 false를 설정합니다.

값 반환

없습니다.

주의

기본적으로 테스트된 EA에서 생성된 모든 지표는 시각적 테스트 차트에 표시됩니다. 또한 이러한 지표는 테스트가 완료되면 자동으로 열리는 차트에 표시됩니다. TesterHideIndicators() 기능을 통해 개발자는 사용된 지표의 표시를 비활성화하는 기능을 규현할 수 있습니다.

EA 테스트 시 적용된 지표 표시를 비활성화 하려면 EA 핸들을 만들기 전에 TesterHideIndicators()를 true로 호출하고, 그 후에 생성된 모든 지표는 숨기기 플래그로 표시됩니다. 이러한 짚표는 비주얼 테스트 도중 또는 테스트가 완료되면 자동으로 열리는 차트에 표시되지 않습니다.

새로 생성된 지표의 숨기기 모드를 비활성화 하려면 TesterHideIndicators()를 false로 호출합니다. 테스트된 EA에서 직접 생성된 지표만 테스트 차트에 표시할 수 있습니다. 이 규칙은 <data_folder>MQL5₩Profiles₩Templates에 탬플릿이 하나만 없는 경우에만 적용됩니다.

<data_folder>MQL5₩Profiles₩Templates 디렉토리에 특수 템플릿 <EA_name>.tpl이 포함된 경우, 시각적 테스트 도중과 테스트 차트에 이 템플릿의 지표만 표시됩니다. 이 경우, 테스트된 EA에 적용된 지표가 표시되지 않습니다. 이 동작은 true와 동일한 TesterHideIndicators()이 EA 코드에서 호출되도록 유지됩니다.

₩data_folder>MQL5₩Profiles₩Templates 디렉토리에 특수 <EA_name>.tpl 템플릿이 없는 경우 tester.tpl의 지표 및 TesterHideIndicators() 기능에 의해 비활성화 되지 않은 EA 지표는 비주얼 테스트 동안 그리고 테스트 차트에 표시됩니다. tester.tpl 템플릿이 없는 경우, default.tpl 템플릿의 지표가 대신 사용됩니다.

전략 테스터가 적절한 템플릿 (<EA_name>.tpl, tester.tpl 또는 default.tpl)을 찾지 못한 경우, EA에 적용된 지표의 디스플레이는 TesterHideIndicators() 기능에 완전히 관리됩니다.

예를 들어:

```
bool CSampleExpert::InitIndicators(void)
  {
   TesterHideIndicators(true);
//--- MACD 지표 생성
   if(m_handle_macd==INVALID_HANDLE)
      if((m_handle_macd=iMACD(NULL,0,12,26,9,PRICE_CLOSE))==INVALID_HANDLE)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         printf("Error creating MACD indicator");
         return(false);
        }
   TesterHideIndicators(false);
//--- EMA 지표 생하여 이를 콜렉션에 추가
   if(m_handle_ema==INVALID_HANDLE)
      if((m_handle_ema=iMA(NULL,0,InpMATrendPeriod,0,MODE_EMA,PRICE_CLOSE))==INVALID_H
        {
         printf("EMA 지표 생성 오류");
         return(false);
        }
//--- 성공
   return(true);
  }
```

추가 참조

IndicatorRelease

© 2000-2025, MetaQuotes Ltd.

TesterStatistics

이 함수는 검사 결과를 기반으로 계산된 지정된 통계 매개변수의 값을 반환.

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // ID
   );
```

매개변수

statistic_id

[in] ENUM_STATISTICS 열거에서 가져온 통계 매개변수의 ID.

값 반환

검사 결과의 통계 매개변수 값.

주의

이 기능은 테스터의 OnTester() 또는 OnDeinit() 내에서 호출할 수 있습니다. 다른 경우에는 결과가 정의되지 않습니다.

예:

```
// 표준 "MACD Sample.mq5" 파일을 기반으로 한 EA
// Tester 이벤트 핸들러에서 TesterStatistics()의 결과를 표시합니다.
#define MACD_MAGIC 1234502
//---
#include <Trade\Trade.mqh>
#include <Trade\SymbolInfo.mqh>
#include <Trade\PositionInfo.mqh>
#include <Trade\AccountInfo.mqh>
//---
input double InpLots          =0.1; // 랏
input int    InpTakeProfit    =50;  // 수익(핍)
input int    InpTrailingStop  =30;  // 트레일링 스탑 레벨(핍)
input int    InpMACDOpenLevel =3;   // MACD 오픈 레벨(핍)
input int    InpMACDCloseLevel=2;   // MACD 클로즈 레벨(핍)
input int    InpMATrendPeriod =26;  // MA 트렌드 기간
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- 필요한 모든 객체를 생성합니다.
   if(!ExtExpert.Init())
      return(INIT_FAILED);
//--- 초기화 성공
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert new tick handling function                                |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   static datetime limit_time=0; // 'timeout'을 고려한 마지막 호출 시간
//--- if the time exceeds the specified limit_time value
   if(TimeCurrent()>=limit_time)
     {
      //--- 데이터 확인
      if(Bars(Symbol(),Period())>2*InpMATrendPeriod)
        {
         //--- 성공하면 Limit_time을 'timeout'초만큼 늘립니다.
         if(ExtExpert.Processing())
            limit_time=TimeCurrent()+ExtTimeOut;
        }
     }
  }
//+------------------------------------------------------------------+
//| Expert tester handling function                                  |
//+------------------------------------------------------------------+
double OnTester(void)
  {
   double ret=TesterStatistics(STAT_PROFIT_FACTOR);
   double profit=TesterStatistics(STAT_PROFIT);
   int    trades_total=(int)TesterStatistics(STAT_TRADES);
   int    profit_total=(int)TesterStatistics(STAT_PROFIT_TRADES);
   int    loss_total=(int)TesterStatistics(STAT_LOSS_TRADES);
   PrintFormat("%s: Profit = %.2f, trades total: %lu, profit trades total: %lu, loss t
   return(ret);
   /*
   Result:
   OnTester: Profit = 209.84, trades total: 13, profit trades total: 11, loss trades t
   final balance 10209.84 USD
   OnTester result 3.020606644198363
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TesterStop

프로그램 작업 명령은 테스트할 때 제공합니다.

```
void  TesterStop();
```

값 반환

반환 값 없음.

주의

TesterStop() 기능은 예를 들어 지정된 수의 손실 거래 또는 사전 설정된 인출 수준에 도달할 때 테스트 에이전트에서 EA를 정기적으로 조기 종료하도록 고안되었습니다.

TesterStop() 호출은 정상적인 테스트 완료로 간주되므로 OnTester() 함수가 호출되고 전체 누적 거래 통계 및 최적화 기준 값이 전략 테스터에 제출됩니다.

전략 테스터에서 ExpertRemove()을 호출하는 것은 정상적인 테스트 완료를 의미하며 거래 통계를 얻을 수 있지만 EA는 에이전트의 메모리에서 언로드 됩니다. 이 경우 다음 매개변수 집합에서 패스를 수행하려면 프로그램을 다시 로드하는 데 시간이 필요합니다. 따라서 TesterStop()는 테스트의 조기 완료를 위한 기본 옵션입니다.

예:

```
//--- 정의
#define BALANCE_LOSS_STOP  100.0       // 테스트가 중단되는 잔액 감소 값
#define EQUITY_LOSS_STOP   100.0       // 테스트가 중단되는 실시간 잔고 감소 가치

//--- 입력 매개변수
input  double  InpLots        =  0.1;  // 랏
input  uint    InpStopLoss    =  50;   // 스탑로스 포인트로
input  uint    InpTakeProfit  =  150;  // 이익 실현 포인트로
sinput ulong   InpMagic       =  123;  // 매직 넘버
sinput ulong   InpDeviation   =  5;    // 편차
//--- 글로벌 변수
CTrade      trade;                     // 거래 클래스 인스턴스
CSymbolInfo symb;                      // 심볼 클래스 인스턴스
CAccountInfo account;                  // 거래 계정 클래스 인스턴스
...
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- 초기화 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 현재 호가 업데이트
   if(!symb.RefreshRates())
      return;
   ...

//--- 잔액이나 실시간 잔고사 BALANCE_LOSS_STOP 및 EQUITY_LOSS_STOP 매크로 대체에 표시된 것보
//--- 테스트가 실패한 것으로 간주되고 TesterStop() 함수가 호출됩니다.
//--- BALANCE_LOSS_STOP 이상으로 잔액이 손실되었는지 확인합니다.
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()<balance_prev-BALANCE_LOSS_STOP)
        {
         PrintFormat("The initial balance of %.2f %s decreased by %.2f %s, and now has
         TesterStop();
         /*
         Result:
         The initial balance of 10000.00 USD decreased by 100.10 USD, and now has a va
         TesterStop() called on 9% of testing interval
         */
        }
     }
//--- EQUITY_LOSS_STOP 이상으로 실시간 잔고의 손실을 확인합니다.
   if(equity_prev!=account.Equity())
     {
      if(account.Equity()<equity_prev-EQUITY_LOSS_STOP)
        {
         PrintFormat("The initial equity of %.2f %s decreased by %.2f %s, and now has
         TesterStop();
         /*
         Result:
         The initial equity of 10000.00 USD decreased by 100.10 USD, and now has a val
         TesterStop() called on 9% of testing interval
         */
        }
     }
  }
```

추가 참조

프로그램 실행, 트레이딩 전략 테스트, ExpertRemove, SetReturnError

© 2000-2025, MetaQuotes Ltd.

TesterDeposit

테스트 중에 예금 자금을 모방하는 특수 기능. 일부 자금 관리 시스템에 사용될 수 있습니다.

```
bool  TesterDeposit(
   double money      // 예치된 합계
   );
```

매개변수

돈

[in] 예금 통화로 계좌에 입금할 돈

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

예:

```
//--- 정의
#define BALANCE_LOSS_DEPOSIT  100.0    // 자금이 테스터의 계좌에 입금될 잔액 인출 값

//--- 입력 매개변수
input  double  InpLots         =  0.1;  // 랏
input  uint    InpStopLoss     =  50;   // 스탑로스 포인트로
input  uint    InpTakeProfit   =  150;  // 이익 실현 포인트로
sinput ulong   InpMagic        =  123;  // 매직 넘버
sinput ulong   InpDeviation    =  5;    // 편차
//--- 글로벌 변수
CTrade      trade;                      // 거래 클래스 인스턴스
CSymbolInfo symb;                       // 심볼 클래스 인스턴스
CAccountInfo account;                   // 거래 계정 클래스 인스턴스
...
double      balance_dep_summ;           // 잔액 충전 총액
uint        balance_dep_total;          // 잔액 충전 횟수
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- 초기 잔액 값을 저장합니다.
   balance_prev=account.Balance();
   balance_dep_summ=0;
   balance_dep_total=0;
//--- 초기화 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 현재 호가 업데이트
   if(!symb.RefreshRates())
      return;
   ...

//--- 잔액이 BALANCE_LOSS_DEPOSIT 매크로 대체에 설정된 것보다 더 많이 떨어진 경우,
//--- 계정을 충전하고 TesterDeposit() 함수를 호출해야 합니다.
//--- BALANCE_LOSS_DEPOSIT 이상의 잔액 손실을 확인
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()<balance_prev-BALANCE_LOSS_DEPOSIT)
        {
         double loss=balance_prev-account.Balance();
         PrintFormat("The initial balance of %.2f %s decreased by %.2f %s. It is neces
         if(TesterDeposit(loss))
           {
            balance_dep_total++;
            balance_dep_summ+=loss;
            balance_prev=account.Balance();
            PrintFormat("Funds have been deposited into the account. Account balance:
            PrintFormat("Total deposits: %lu. Amount of deposits: %.2f %s.",balance_de
           }
         /*
         Result:
         The initial balance of 10000.00 USD decreased by 116.00 USD. It is necessary
         deal #45 balance 116.00 [deposit] done
         Funds have been deposited into the account. Account balance: 10000.00 USD.
         Total deposits: 1. Amount of deposits: 116.00 USD.
         */
        }
     }
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- 재무적 측면에서 최대 잔액 감소를 출력 핸들러 값으로 설정합니다.
   double ret=TesterStatistics(STAT_BALANCE_DD);
//--- 인출, 예금 수 및 총 금액에 대한 메시지를 로그에 표시합니다.
   PrintFormat("%s: Maximum balance drawdown in money: %.2f %s. Total deposits: %lu. A
//--- 결과 반환
   return(ret);
   /*
   Result:
   OnTester: Maximum balance drawdown in money: 5188.50 USD. Total deposits: 46. Amour
```

© 2000-2025, MetaQuotes Ltd.

```
   final balance 4867.50 USD
   OnTester result 5188.5
   */
  }
```

추가 참조

TesterWithdrawal

© 2000-2025, MetaQuotes Ltd.

TesterWithdrawal

테스트 과정에서 자금 인출의 운용을 모방하는 특별한 기능. 일부 자산 관리 시스템에서 사용할 수 있습니다.

```
bool  TesterWithdrawal(
   double money      // 인출하려는 금액
   );
```

매개변수

돈

[in] 출금해야 하는 금액(예금 통화 중)

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

예:

```
//--- 정의
#define BALANCE_PROFIT_WITHDRAWAL   5  // 테스터의 계좌에서 자금이 인출될 때 잔액 수익의 가

//--- 입력 매개변수
input  double  InpLots        =  0.1;  // 랏
input  uint    InpStopLoss    =  50;   // 스탑로스 포인트로
input  uint    InpTakeProfit  =  150;  // 이익 실현 포인트로
sinput ulong   InpMagic       =  123;  // 매직 넘버
sinput ulong   InpDeviation   =  5;    // 편차
//--- 글로벌 변수
CTrade      trade;                     // 거래 클래스 인스턴스
CSymbolInfo symb;                      // 심볼 클래스 인스턴스
CAccountInfo account;                  // 거래 계정 클래스 인스턴스
...
double      balance_op_sum;            // 잔액 작업의 총량
uint        balance_op_total;          // 잔액 작업의 횟수
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
   ...
//--- 초기 잔액 값을 저장합니다.
   balance_prev=account.Balance();
   balance_op_sum=0;
   balance_op_total=0;
//--- 초기화 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 현재 호가 업데이트
   if(!symb.RefreshRates())
      return;
   ...

//--- 잔액 수익이 BALANCE_PROFIT_WITHDRAWAL 매크로 대체에 지정된 값만큼 현재 잔액을 초과하는
//--- 이 자금을 계좌에서 인출해야 합니다. Call the TesterWithdrawal() function.
//--- BALANCE_PROFIT_WITHDRAWAL 초과에 대한 잔액 수익 확인
   if(balance_prev!=account.Balance())
     {
      if(account.Balance()>balance_prev+BALANCE_PROFIT_WITHDRAWAL)
        {
         double profit=account.Balance()-balance_prev;
         PrintFormat("The account balance has been increased by %.2f %s. Need to withd
         if(TesterWithdrawal(profit))
           {
            balance_op_total++;
            balance_op_summ+=profit;
            balance_prev=account.Balance();
            PrintFormat("Funds have been withdrawn from the account. Account balance:
            PrintFormat("Total withdrawals: %lu. Amount of withdrawals: %.2f %s.",bala
           }
         /*
         Result:
         The account balance has been increased by 21.00 USD. Need to withdraw these f
         deal #13 balance -21.00 [withdrawal] done
         Funds have been withdrawn from the account. Account balance: 10000.00 USD.
         Total withdrawals: 1. Amount of withdrawals: 21.00 USD.
         */
        }
     }
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- 재무적 측면에서 최대 잔액 감소를 출력 핸들러 값으로 설정합니다.
   double ret=TesterStatistics(STAT_BALANCE_DD);
//--- 인출 횟수, 총 인출 금액 및 손실 대한 메시지를 로그에 표시합니다.
   PrintFormat("%s: Maximum balance drawdown in money: %.2f %s. Total withdrawals: %lu
//--- 결과 반환
   return(ret);
   /*
   Result:
```

© 2000-2025, MetaQuotes Ltd.

```
   OnTester: Maximum balance drawdown in money: 5188.50 USD. Total withdrawals: 2. Amo
   final balance 4867.50 USD
   OnTester result 5188.5
   */
  }
```

추가 참조

TesterDeposit

© 2000-2025, MetaQuotes Ltd.

TranslateKey

현재 입력 언어와 제어 키 상태를 고려하여 가상 키 코드로 유니코드 문자를 반환합니다.

```
short  TranslateKey(
   int  key_code      // 유니코드 문자를 수신하기 위한 키 코드
   );
```

매개변수

key_code

[in] 키 코드.

값 반환

변환에 성공한 경우 유니코드 문자입니다. 이 함수는 오류 발생시 -1을 반환합니다.

주의

이 함수는 ToUnicodeEx를 사용하여 사용자가 누르는 키를 유니코드 문자로 변환합니다. ToUnicodeEx가 트리거 되지 않은 경우, 예를 들어 SHIFT 키 문자를 수신하려고 할 때 오류가 발생할 수 있습니다.

예를 들어:

```
void OnChartEvent(const int id,const long& lparam,const double& dparam,const string& s
  {
   if(id==CHARTEVENT_KEYDOWN)
     {
      short sym=TranslateKey((int)lparam);
      //--- 입력한 문자가 유니코드로 변환된 경우
      if(sym>0)
         Print(sym,"'",ShortToString(sym),"'");
      else
         Print("Error in TranslateKey for key=",lparam);
     }
  }
```

추가 참조

클라이언트 터미널 이벤트, OnChartEvent

© 2000-2025, MetaQuotes Ltd.

ZeroMemory

이 함수는 참조로 전달된 변수를 재설정.

```
void  ZeroMemory(
   void & variable      // 변수를 재설정
   );
```

매개변수

변수

[in] [out] 참조로 전달된 변수를 재설정합니다(영점 값으로 초기화)

값 반환

반환 값 없음.

주의

함수 매개변수가 문자열인 경우 호출은 NULL을 값으로 표시하는 것과 같습니다.
단순 유형 및 해당 문자열뿐 아니라 이러한 유형으로 구성된 구조/클래스의 경우 이는 간단한 재설정입니다.
문자열과 동적 문자열을 포함하는 개체의 경우 각 요소에 대해 ZeroMemory()가 호출됩니다.
수식어로 보호되지 않는 문자열의 경우 이는 모든 요소의 영점 조정입니다.
복잡한 개체 문자열의 경우 각 요소에 대해 ZeroMemory()가 호출됩니다.

ZeroMemory()는 보호된 멤버 또는 계승이 있는 클래스에 적용할 수 없습니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 문자열 선언 및 초기화
   string str="Test ZeroMemory func";
//--- ZeroMemory()를 적용하기 전에 로그에 행을 보냅니다.
   PrintFormat("The line before applying ZeroMemory() to it: '%s'",str);
//--- 문자열을 재설정하고 결과를 로그로 보냅니다.
   ZeroMemory(str);
   Print("The same line after applying ZeroMemory() to it: '",str,"'");
  /*
   Result:
   The line before applying ZeroMemory() to it: 'Test ZeroMemory func'
   The same line after applying ZeroMemory() to it: ''
  */

//--- int형 변수 선언 및 초기화
   int var=123;
//--- ZeroMemory()를 적용하기 전에 로그에 행을 보냅니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("\nThe integer variable before applying ZeroMemory() to it: %d",var);
//--- 변수를 재설정하고 결과를 로그에 보냅니다.
   ZeroMemory(var);
   PrintFormat("The same variable after applying ZeroMemory() to it: %d",var);
  /*
   Result:
   The integer variable before applying ZeroMemory() to it: 123
   The same variable after applying ZeroMemory() to it: 0
  */

//--- int형 배열을 선언하고 초기화합니다.
   int arr[]={0,1,2,3,4,5,6,7,8,9};
//--- ZeroMemory()를 적용하기 전에 배열을 로그로 보냅니다.
   Print("\nThe integer array before applying ZeroMemory() to it:");
   ArrayPrint(arr);
//--- 배열을 재설정하고 결과를 로그에 보냅니다.
   ZeroMemory(arr);
   Print("The same array after applying ZeroMemory() to it:");
   ArrayPrint(arr);
  /*
   Result:
   The integer array before applying ZeroMemory() to it:
   0 1 2 3 4 5 6 7 8 9
   The same array after applying ZeroMemory() to it:
   0 0 0 0 0 0 0 0 0 0
  */

//--- 두 필드(문자열 및 정수 필드)의 구조를 선언합니다.
   struct STest
     {
      string   var_string;
      long     var_long;
     };
//--- STest 구조 유형의 배열을 선언하고 초기화합니다.
   STest arr_struct[]={ {"0",0}, {"1",1}, {"2",2}, {"3",3} };
//--- ZeroMemory()를 적용하기 전에 배열을 로그로 보냅니다.
   Print("\nThe array struct before applying ZeroMemory() to it:");
   ArrayPrint(arr_struct);
//--- 구조 배열을 재설정하고 결과를 로그로 보냅니다.
   ZeroMemory(arr_struct);
   Print("The same array struct after applying ZeroMemory() to it:");
   ArrayPrint(arr_struct);
  /*
   Result:
   The array struct before applying ZeroMemory() to it:
       [var_string] [var_long]
   [0] "0"                   0
   [1] "1"                   1
   [2] "2"                   2
```

© 2000-2025, MetaQuotes Ltd.

```
   [3] "3"                   3
   The same array struct after applying ZeroMemory() to it:
       [var_string] [var_long]
   [0] null                  0
   [1] null                  0
   [2] null                  0
   [3] null                  0
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

배열 작업을 위한 함수 그룹

배열은 최대 4차원이 가능합니다. 각 차원은 0에서 *dimension_size-1*로 인덱싱 됩니다. 50개 요소의 1차원 배열의 경우, 첫번째 요소를 호출하면 마지막 요소의 배열[0](배열[49])로 표시됩니다.

기능	액션
ArrayBsearch	첫 번째 배열 차원에 있는 첫 번째 발견된 요소의 인덱스를 반환
ArrayCopy	한개 배열을 다른 배열로 복사
ArrayCompare	다음 두 배열을 비교한 결과를 반환:단순 유형 또는 맞춤형 구조 복합 개체
ArrayFree	모든 동적 배열의 버퍼를 해제하고 0 차원의 크기를 0으로 설정
ArrayGetAsSeries	배열 인덱싱 방향을 확인
ArrayInitialize	수열의 모든 요소를 단일 값으로 설정
ArrayFill	지정된 값으로 배열을 채우기
ArrayIsSeries	배열이 시계열인지 아닌지 확인
ArrayIsDynamic	배열이 동적인지 아닌지 확인
ArrayMaximum	최대값으로 요소 검색
ArrayMinimum	최소값으로 요소 검색
ArrayPrint	단순 유형 또는 단순 구조의 배열을 저널에 인쇄
ArrayRange	배열의 지정된 차원에 있는 요소의 수를 반환
ArrayResize	베열의 첫 번째 차원에서 새로 크기를 설정
ArrayInsert	소스 배열에서 지정된 수의 요소를 지정된 인덱스에서 시작하여 수신하는 요소에 삽입
ArrayRemove	지정된 인덱스로 시작하는 배열에서 지정된 수의 요소를 제거
ArrayReverse	지정된 인덱스로 시작하는 배열에서 지정된 수의 요소를 반전
ArraySetAsSeries	배열 인텍싱의 방향을 설정
ArraySize	배열에서 요소의 수를 반환
ArraySort	첫 번째 차원을 기준으로 수열 정렬
ArraySwap	동일 유형의 두 동적 배열 내용을 스왑
ArrayToFP16	float 또는 double 유형의 배열을 주어진 형식의 ushort 유형의 배열에 복사합니다
ArrayToFP8	float 또는 double 유형의 배열을 지정된 형식의 uchar 유형의 배열에 복사합니다
ArrayFromFP16	ushort 유형의 배열을 주어진 형식의 float 또는 double 유형의 배열로 복사합니다

© 2000-2025, MetaQuotes Ltd.

기능	액션
ArrayFromFP8	uchar 유형의 배열을 지정된 형식의 float 또는 double 유형의 배열로 복사합니다

© 2000-2025, MetaQuotes Ltd.

ArrayBsearch

오름차순으로 정렬된 다차원 숫자 배열에서 지정된 값을 검색. 검색은 첫 번째 차원의 요소를 통해 수행됩니다.

double 유형의 배열에서 검색하는 경우

```
int  ArrayBsearch(
   const double&    array[],   // 검색할 배열
   double           value      // 찾고 있는 것
   );
```

float 유형의 배열에서 검색하는 경우

```
int  ArrayBsearch(
   const float&    array[],   // 검색할 배열
   float           value      // 찾고 있는 것
   );
```

long 유형의 배열에서 검색하는 경우

```
int  ArrayBsearch(
   const long&    array[],   // 검색할 배열
   long           value      // 찾고 있는 것
   );
```

int 유형의 배열에서 검색하는 경우

```
int  ArrayBsearch(
   const int&    array[],   // 검색할 배열
   int           value      // 찾고 있는 것
   );
```

short 유형 배열에서 검색하는 경우

```
int  ArrayBsearch(
   const short&    array[],   // 검색할 배열
   short           value      // 찾고 있는 것
   );
```

char 유형의 배열에서 검색하는 경우

```
int  ArrayBsearch(
   const char&    array[],   // 검색할 배열
   char           value      // 찾고 있는 것
   );
```

매개변수

array[]

[in] 검색할 숫자 배열.

값

© 2000-2025, MetaQuotes Ltd.

[in] 검색 값.

반환 값

함수는 발견된 요소의 인덱스를 반환합니다. 원하는 값을 찾을 수 없으면 함수는 값에 가장 가까운 요소의 인덱스를 반환합니다.

참고

바이너리 검색 프로세스는 정렬된 배열을 처리합니다. 숫자 배열을 정렬하려면 ArraySort() 기능을 사용.

예:

```
#property description "RSI 지표 데이터 표시 기준 스크립트"
#property description "시장이 얼마나 자주 열렸는지"
#property description "지정된 시간 간격 동안 과잉구매 및 과잉판매된 영역."
//--- 스크립트를 시작할 때 입력 매개변수 창 표시
#property script_show_inputs
//--- 매개변수 입력
input int                InpMAPeriod=14;                  // 이동 평균 주기
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;     // 가격 유형
input double             InpOversoldValue=30.0;           // 과잉판매 레벨
input double             InpOverboughtValue=70.0;         // 과잉구매 레벨
input datetime           InpDateStart=D'2012.01.01 00:00'; // 분석 시작일
input datetime           InpDateFinish=D'2013.01.01 00:00'; // 분석 종료일
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   double rsi_buff[]; // 지표 값 배열
   int    size=0;     // 배열 크기
//--- RSI 지표 핸들 수신
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- 지표 핸들을 수신하는데 실패
      PrintFormat("지표 핸들 수신 오류. 오류 코드 = %d",GetLastError());
      return;
     }
//--- 지표가 모든 값을 계산할 때까지 루프에 있음
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- 지표가 스크립트 작업을 강제로 완료한 경우 종료
      if(IsStopped())
         return;
      //--- 지표가 모든 값을 계산할 수 있도록 잠시 멈춤
      Sleep(10);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 일정 기간 동안 지표 값을 복사
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("지표 값을 복사하는데 실패. 오류 코드 = %d",GetLastError());
      return;
     }
//--- 배열 크기 수신
   size=ArraySize(rsi_buff);
//--- 배열을 정리
   ArraySort(rsi_buff);
//--- 시장이 과잉판매된 시간(백분율)에 대해 확인
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- 시장이 과잉구매된 시간(백분율)에 대해 확인
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)siz
//--- 데이터를 표시하기 위한 문자열을 구성
   string str="From "+TimeToString(InpDateStart,TIME_DATE)+" to "
              +TimeToString(InpDateFinish,TIME_DATE)+" the market was:";
   string str_ovb="in overbought area "+DoubleToString(ovb,2)+"% of time";
   string str_ovs="in oversold area "+DoubleToString(ovs,2)+"% of time";
//--- 데이터를 차트에 표시
   CreateLabel("top",5,60,str,clrDodgerBlue);
   CreateLabel("overbought",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("oversold",5,10,str_ovs,clrDodgerBlue);
//--- 차드 다시 그리기
   ChartRedraw(0);
//--- 일시 정지
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| 차트 좌측 하단 모서리에 설명 표시                                |
//+------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- 라벨 생성
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- 라벨을 좌측 하단 모서리에 바인딩
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- 고정점의 위치를 변경
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- X 방향으로 고정점으로부터의 거리
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- Y 방향으로 고정점으로부터의 거리
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- 라벨 텍스트
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- 텍스트 색상
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 텍스트 크기
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayCopy

배열을 다른 배열로 복사합니다.

```
int  ArrayCopy(
   void&        dst_array[],          // 대상 배열
   const void&  src_array[],          // 원본 배열
   int          dst_start=0,          // 대상 배열에 쓸 인덱스 시작
   int          src_start=0,          // 원본 배열의 첫 번째 인덱스
   int          count=WHOLE_ARRAY     // 요소의 수
   );
```

매개변수

dst_array[]

[out] 대상 배열

src_array[]

[in] 원본 배열

dst_start=0

[in] 대상 배열에서의 시작 인덱스. 기본적으로 시작 인덱스는 0입니다.

src_start=0

[in] 원본 배열의 시작 인덱스. 기본적으로 시작 인덱스는 0입니다.

count=WHOLE_ARRAY

[in] 복사할 요소의 수. 기본적으로 전체 배열이 복사됩니다 (count=WHOLE_ARRAY).

반환 값

복사된 요소의 수를 반환합니다.

참고

count<0 또는 count>src_size-src_start이면, 나머지 배열 부분이 모두 복사됩니다. 배열이 좌측에서 우측으로 복사됩니다. 배열 시리즈의 경우, 좌측에서 우측으로 복사하기 위해 시작 포지션이 올바르게 정의됩니다.

배열 유형이 다른 경우 복사하는 동안 원본 배열의 각 요소를 대상 배열 유형으로의 변환을 시도합니다. 문자열 배열은 문자열 배열로만 복사할 수 있습니다. 초기화가 필요하지 않은 개체를 포함하는 클래스 및 구조의 배열은 복사되지 않습니다. 구조 배열을 동일한 유형의 배열로만 복사할 수 있습니다.

For dynamic arrays with indexing as in 시계열과 같은 인덱싱을 사용하는 동적 배열의 경우 대상 배열 크기가 복사된 데이터의 양(배열의 크기를 초과하는 경우)으로 자동으로 증가합니다. 대상 배열 크기는 자동으로 줄어들지 않습니다.

예:

```
#property description "지표가 로컬 캔들스틱을 강조처리 합니다"
#property description "고가 및 저가. 검색 간격 길이"
#property description "입력 매개변수를 사용하여 극한값을 찾아야 합니다."
//--- 지표 설정
#property indicator_chart_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 5
#property indicator_plots   1
//---- 플롯
#property indicator_label1  "극값"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrLightSteelBlue,clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 사전정의된 상수
#define INDICATOR_EMPTY_VALUE 0.0
//--- 매개변수 입력
input int InpNum=4; // 간격 길이의 반
//--- 지표 버퍼
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- 글로벌 변수
int    ExtStart=0; // 극값이 아닌 최초의 캔들스틱 인덱스
int    ExtCount=0; // 간격 내 비극값 수
//+------------------------------------------------------------------+
//| 비극값 캔들 스틱 채움                                            |
//+------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- 캔들스틱 채움
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- 표시되지 않ㅇ는 값을 지정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- 데이터 창에 표시할 지표 버퍼 이름 지정
   PlotIndexSetString(0,PLOT_LABEL,"시가; 고가; 저가; 종가");
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 시계열에서 정확한 인덱스 설정
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- 막대 계산 시작의 변수
   int start=prev_calculated;
//--- 첫 번째 InpNum*2 막대에 대한 계산이 수행되지 않습니다
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//--- 막대가 막 형성된 경우, 다음 극값을 확인
   if(rates_total-start==1)
      start--;
//--- 극값에 대해 확인할 막대 인덱스
   int ext;
//--- 지표 값 계산 루프
   for(int i=start;i<rates_total-1;i++)
     {
      //--- 초기에 그려지지 않은 i 막대
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- 확인할 극값 인덱스
      ext=i-InpNum;
      //--- 로컬 최대값 확인
      if(IsMax(high,ext))
        {
         //--- 극값의 캔들스틱 강조
```

© 2000-2025, MetaQuotes Ltd.

```
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //--- 다른 캔들스틱을 극단까지 뉴트럴 색상으로 강조
         FillCandles(open,high,low,close);
         //--- 변수 색상 변경
         ExtStart=ext+1;
         ExtCount=0;
         //--- 다음 반복으로 넘어가기
         continue;
        }
      //--- 로컬 최소값 확인
      if(IsMin(low,ext))
        {
         //--- 극값의 캔들스틱 강조
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //--- 다른 캔들스틱을 극단까지 뉴트럴 색상으로 강조
         FillCandles(open,high,low,close);
         //--- 변수 값 변경
         ExtStart=ext+1;
         ExtCount=0;
         //--- 다음 반복으로 넘어가기
         continue;
        }
      //--- 간격에 따라 비극값 수를 늘림
      ExtCount++;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 현재 배열 요소가 로컬 고가인지 확인                              |
//+------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- 간격 시작 변수
   int i=ind-InpNum;
//--- 간격 종료 기간
   int finish=ind+InpNum+1;
//--- 간격의 전반부를 점검
   for(;i<ind;i++)
     {
      if(price[ind]<=price[i])
```

© 2000-2025, MetaQuotes Ltd.

```
         return(false);
      }
//--- 간격의 후반부를 점검
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]<=price[i])
         return(false);
      }
//--- 이는 극값입니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 현재 배열 요소가 로컬 저가인지 확인                              |
//+------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- 간격 시작 변수
   int i=ind-InpNum;
//--- 간격 끝 변수
   int finish=ind+InpNum+1;
//--- 간격의 전반부를 점검
   for(;i<ind;i++)
     {
      if(price[ind]>=price[i])
         return(false);
      }
//--- 간격의 후반부를 점검
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]>=price[i])
         return(false);
      }
//--- 이는 극값입니다
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayCompare

함수는 동일한 두 배열을 비교한 결과를 반환. 간단한 유형 또는 복잡한 개체가 없는 구조를 비교하는데 사용할 수 있으며, 이는 사용자 지정 구조로서 문자열, 동적 배열, 클래스 및 기타 복잡한 개체를 가진 구조를 포함하지 않습니다.

```
int  ArrayCompare(
   const void&  array1[],           // 첫 번째 배열
   const void&  array2[],           // 두 번째 배열
   int          start1=0,           // 첫 번째 배열의 초기 오프셋
   int          start2=0,           // 두 번째 배열의 초기 오프셋
   int          count=WHOLE_ARRAY   // 비교할 요소의 수
   );
```

매개변수

array1[]

[in] 첫 번째 배열.

array2[]

[in] 두 번째 배열.

start1=0

[in] 첫 번째 배열에서 비교가 시작되는 요소의 초기 인덱스. 기본 시작 인덱스 -0.

start2=0

[in] 비교가 시작되는 두 번째 배열에 있는 요소의 초기 인덱스. 기본 시작 인덱스 -0.

count=WHOLE_ARRAY

[in] 비교할 요소의 수. 기본적으로 두 배열의 요소가 비교됩니다 (count=WHOLE_ARRAY).

반환 값

- array1[]이 array2[]보다 작으면 -1
- array1[]이 array2[]와 같으면 0
- array1[]이 array2[]보다 크면 1
- 비교 배열 유형이 호환되지 않아 오류가 발생하거나 start1, start2 또는 카운트 값이 배열 외부에 속하는 경우 -2.

참고

배열의 크기가 다르면 0(배열은 동일한 것으로 간주되지 않음)을 반환하지 않으며, 배열이 다른 배열의 신뢰할 수 있는 하위 집합인 경우 count=WHOLE_ARRAY를 반환합니다. 이 경우 해당 배열의 크기를 비교한 결과가 반환되며, -1은 array1[] 사이즈가 array2[] 사이즈보다 작은 경우, 1은 그렇지 않은 경우입니다.

예:

```
//--- 글로벌 변수
double   ExtArrayFirst[];
double   ExtArraySecond[];
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 배열 크기를 정합니다
   if(ArrayResize(ExtArrayFirst,10)!=10)
     {
      Print("ArrayResize() failed for ExtArrayFirst. Error code: ",GetLastError());
      return;
     }
   if(ArrayResize(ExtArraySecond,10)!=10)
     {
      Print("ArrayResize() failed for ExtArraySecond. Error code: ",GetLastError());
      return;
     }

//--- 루프에서 i 및 j 인덱스 값으로 배열을 채웁니다.
   int total=ArraySize(ExtArrayFirst);
   for(int i=0, j=total-1; i<total; i++,j--)
     {
      //--- ExtArrayFirst 배열을 왼쪽에서 오른쪽으로 채웁니다.
      //--- 오른쪽에서 왼쪽으로 ExtArraySecond 배열을 채웁니다.
      ExtArrayFirst[i]=i;
      ExtArraySecond[i]=j;
     }
//--- 배열을 비교하고 결과를 로그에 인쇄합니다.
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
   Result:
   ExtArrayFirst:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   ExtArraySecond:
   9.00000 8.00000 7.00000 6.00000 5.00000 4.00000 3.00000 2.00000 1.00000 0.00000
   Result ArrayCompare(): ExtArrayFirst is smaller than ExtArraySecond (result = -1)
   */

//--- 이제 배열을 뒤집어 보겠습니다.
//--- 루프에서 i 및 j 인덱스 값으로 배열을 채웁니다.
   for(int i=0, j=total-1; i<total; i++,j--)
     {
      //--- 오른쪽에서 왼쪽으로 ExtArrayFirst 배열을 채웁니다.
      //--- ExtArraySecond 배열을 왼쪽에서 오른쪽으로 채웁니다.
      ExtArrayFirst[i]=j;
      ExtArraySecond[i]=i;
     }
//--- 배열을 비교하고 결과를 로그에 인쇄합니다.
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
```

© 2000-2025, MetaQuotes Ltd.

```
   Result:
   ExtArrayFirst:
   9.00000 8.00000 7.00000 6.00000 5.00000 4.00000 3.00000 2.00000 1.00000 0.00000
   ExtArraySecond:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   Result ArrayCompare(): ExtArrayFirst is larger than ExtArraySecond (result = 1)
   */

//--- 이제 한 방향으로 배열을 채워보겠습니다.
//--- 루프에서 i 인덱스 값으로 배열을 채웁니다.
   for(int i=0; i<total; i++)
     {
      //--- 두 배열을 왼쪽에서 오른쪽으로 채웁니다.
      ExtArrayFirst[i]=i;
      ExtArraySecond[i]=i;
     }
//--- 배열을 비교하고 결과를 로그에 인쇄합니다.
   ArrayComparePrint(ExtArrayFirst,ExtArraySecond);
   /*
   Result:
   ExtArrayFirst:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   ExtArraySecond:
   0.00000 1.00000 2.00000 3.00000 4.00000 5.00000 6.00000 7.00000 8.00000 9.00000
   Result ArrayCompare(): ExtArrayFirst and ExtArraySecond are equal (result = 0)
   */
  }
//+------------------------------------------------------------------+
//| 결과 비교 및 표시                                                |
//+------------------------------------------------------------------+
void ArrayComparePrint(const double &array1[], const double &array2[])
  {
   //--- 배열의 헤더와 내용을 인쇄합니다.
   Print("ExtArrayFirst:");
   ArrayPrint(array1);
   Print("ExtArraySecond:");
   ArrayPrint(array2);
   //--- 배열을 비교하고 비교 결과를 인쇄합니다
   int    res=ArrayCompare(array1,array2);
   string res_str=(res>0 ? "ExtArrayFirst is larger than ExtArraySecond" : res<0 ? "Ex
   PrintFormat("Result ArrayCompare(): %s (result = %d)\n",res_str,res);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

ArrayFree

동적 배열의 버퍼가 해제되고 0차원의 크기가 0으로 설정됩니다.

```
void  ArrayFree(
   void&  array[]      // 배열
   );
```

매개변수

array[]

[in] 동적 배열.

반환 값

반환 값이 없습니다.

참고

사용된 모든 메모리가 한번에 해제되고 배열의 기본 작업이 지표 버퍼에 대한 액세스를 구성하기 때문에 ArrayFree() 기능을 너무 자주 사용할 필요가 없습니다. 버퍼의 크기는 터미널의 실행 하위 시스템에 의해 자동으로 관리됩니다.

애플리케이션의 복잡한 동적 환경에서 메모리를 수동으로 관리해야 하는 경우, ArrayFree() 기능을 사용하면 이미 불필요한 동적 배열이 사용하는 메모리를 명시적이고 즉각적으로 확보할 수 있습니다.

예:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- 사전정의된 상수
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------+
//| 메모리 작업을 위한 대화상자 클래스                                |
//+------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- 배열 크기
   int               m_arr_size;
   //--- 배열
   char              m_arr_char[];
   int               m_arr_int[];
   float             m_arr_float[];
   double            m_arr_double[];
   long              m_arr_long[];
   //--- 라벨
```

© 2000-2025, MetaQuotes Ltd.

```
   CLabel            m_lbl_memory_physical;
   CLabel            m_lbl_memory_total;
   CLabel            m_lbl_memory_available;
   CLabel            m_lbl_memory_used;
   CLabel            m_lbl_array_size;
   CLabel            m_lbl_array_type;
   CLabel            m_lbl_error;
   CLabel            m_lbl_change_type;
   CLabel            m_lbl_add_size;
   //--- 버튼
   CButton           m_button_add;
   CButton           m_button_free;
   //--- 콤보 상자
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- 콤보 상자에서 배열 유형의 현재 값
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- 클래스 개체 생성 방법
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트의 처리기
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- 라벨 생성
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int y
   //--- 제어 요소 생성
   bool              CreateButton(CButton &button,const string name,const int x,const
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- 이벤트 처리기
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- 현재 배열 작업 방법
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------+
//| 현재 배열의 최적화 메모리                                        |
//+------------------------------------------------------------------+
void CMemoryControl::CurrentArrayFree(void)
  {
//--- 배열 크기 재설정
   m_arr_size=0;
//--- 배열 풀기
```

© 2000-2025, MetaQuotes Ltd.

```
   if(m_combo_box_type_value==0)
      ArrayFree(m_arr_char);
   if(m_combo_box_type_value==1)
      ArrayFree(m_arr_int);
   if(m_combo_box_type_value==2)
      ArrayFree(m_arr_float);
   if(m_combo_box_type_value==3)
      ArrayFree(m_arr_double);
   if(m_combo_box_type_value==4)
      ArrayFree(m_arr_long);
  }
//+------------------------------------------------------------------+
//| 현재 배열에 대한 메모리 추가 시도                                |
//+------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//--- 사용된 메모리의 크기가 물리적 메모리의 크기를 초과할 경우 종료
   if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMOR
      return(false);
//--- 현재 유형에 따라 메모리 할당 시도
   if(m_combo_box_type_value==0 && ArrayResize(m_arr_char,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==1 && ArrayResize(m_arr_int,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==2 && ArrayResize(m_arr_float,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==3 && ArrayResize(m_arr_double,m_arr_size)==-1)
      return(false);
   if(m_combo_box_type_value==4 && ArrayResize(m_arr_long,m_arr_size)==-1)
      return(false);
//--- 할당된 메모리
   return(true);
  }
//+------------------------------------------------------------------+
//| 핸들링 이벤트                                                    |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CMemoryControl)
ON_EVENT(ON_CLICK,m_button_add,OnClickButtonAdd)
ON_EVENT(ON_CLICK,m_button_free,OnClickButtonFree)
ON_EVENT(ON_CHANGE,m_combo_box_type,OnChangeComboBoxType)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| 클래스 객체 생성 방법                                            |
//+------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- 기본 클래스 개체 생성
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 라벨을 위한 문자열 준비
   string str_physical="물리적 메모리 = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_P
   string str_total="전체 메모리 = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TOTAL)
   string str_available="사용가능 메모리 = "+(string)TerminalInfoInteger(TERMINAL_MEMORY
   string str_used="사용되는 메모리 = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_USE
//--- 라벨 생성
   if(!CreateLabel(m_lbl_memory_physical,"physical_label",X_START+10,Y_START+5,str_phy
      return(false);
   if(!CreateLabel(m_lbl_memory_total,"total_label",X_START+10,Y_START+30,str_total,12
      return(false);
   if(!CreateLabel(m_lbl_memory_available,"available_label",X_START+10,Y_START+55,str_
      return(false);
   if(!CreateLabel(m_lbl_memory_used,"used_label",X_START+10,Y_START+80,str_used,12,cl
      return(false);
   if(!CreateLabel(m_lbl_array_type,"type_label",X_START+10,Y_START+105,"Array type =
      return(false);
   if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Array size =
      return(false);
   if(!CreateLabel(m_lbl_error,"error_label",X_START+10,Y_START+155,"",12,clrRed))
      return(false);
   if(!CreateLabel(m_lbl_change_type,"change_type_label",X_START+10,Y_START+185,"Chang
      return(false);
   if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Add to arra
      return(false);
//--- 제어 요소 생성
   if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Add",12,clrBlue)
      return(false);
   if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Free",12,clrBl
      return(false);
   if(!CreateComboBoxType())
      return(false);
   if(!CreateComboBoxStep())
      return(false);
//--- 변수 초기화
   m_arr_size=0;
//--- 실행 성공
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 버튼 생성                                                        |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- 버튼 생성
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- 텍스트
   if(!button.Text(str))
      return(false);
//--- 폰트 크기
   if(!button.FontSize(font_size))
      return(false);
//--- 라벨 색상
   if(!button.Color(clr))
      return(false);
//--- 제어 요소에 버튼을 추가
   if(!Add(button))
      return(false);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 배열 크기에 대한 콤보 상자 생성                                  |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- 콤보 상자 생성
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- 콤보 상자에 요소 추가
   if(!m_combo_box_step.ItemAdd("100 000",100000))
      return(false);
   if(!m_combo_box_step.ItemAdd("1 000 000",1000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("10 000 000",10000000))
      return(false);
   if(!m_combo_box_step.ItemAdd("100 000 000",100000000))
      return(false);
//--- 현재 콤보 상자 요소를 설정
   if(!m_combo_box_step.SelectByValue(1000000))
      return(false);
//--- 콤보 상자를 제어 요소에 추가
   if(!Add(m_combo_box_step))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 배열 유형에 대한 콤보 상자 생성                                  |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- 콤보 상자 생성
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- 콤보 상자에 요소 추가
   if(!m_combo_box_type.ItemAdd("char",0))
      return(false);
   if(!m_combo_box_type.ItemAdd("int",1))
      return(false);
   if(!m_combo_box_type.ItemAdd("float",2))
      return(false);
   if(!m_combo_box_type.ItemAdd("double",3))
      return(false);
   if(!m_combo_box_type.ItemAdd("long",4))
      return(false);
//--- 현재 콤보 상자 요소를 설정
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- 현재 콤보 상자 요소를 저장
   m_combo_box_type_value=3;
//--- 콤보 상자를 제어 요소에 추가
   if(!Add(m_combo_box_type))
      return(false);
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 라벨 생성                                                        |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- 라벨 생성
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- 텍스트
   if(!lbl.Text(str))
      return(false);
//--- 폰트 크기
   if(!lbl.FontSize(font_size))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- 색상
   if(!lbl.Color(clr))
      return(false);
//--- 제어 요소에 라벨 추가
   if(!Add(lbl))
      return(false);
//--- 성공함
   return(true);
  }
//+------------------------------------------------------------------+
//| "추가" 버튼 이벤트 클릭 처리기 event                               |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//--- 배열 크기 증가
   m_arr_size+=(int)m_combo_box_step.Value();
//--- 현재 배열에 대해 메모리 할당 시도
   if(CurrentArrayAdd())
     {
      //--- 할당된 메모리, 화면에 현재 상태 표시
      m_lbl_memory_available.Text("사용 가능 메모리 = "+(string)TerminalInfoInteger(TERM
      m_lbl_memory_used.Text("사용되는 메모리 = "+(string)TerminalInfoInteger(TERMINAL_M
      m_lbl_array_size.Text("배열 크기 = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
     }
   else
     {
      //--- 메모리 할당 실패, 오류 메시지 표시
      m_lbl_error.Text("오류 발생, 배열이 너무 큽니다!");
      //--- 이전 배열 크기를 반환
      m_arr_size-=(int)m_combo_box_step.Value();
     }
  }
//+------------------------------------------------------------------+
//| "무료" 버튼 이벤트 처리기                                         |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
  {
//--- 현재 배열의 메모리 최적화
   CurrentArrayFree();
//--- 화면에 현재 상태를 표시
   m_lbl_memory_available.Text("사용 가능한 메모리 = "+(string)TerminalInfoInteger(TERMI
   m_lbl_memory_used.Text("사용되는 메모리 = "+(string)TerminalInfoInteger(TERMINAL_MEMO
   m_lbl_array_size.Text("배열 크기 = 0");
   m_lbl_error.Text("");
  }
//+------------------------------------------------------------------+
//| 콤보 상자 변경 이벤트 처리기                                      |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
  {
//--- 배열 유형이 변경되었는지 확인
   if(m_combo_box_type.Value()!=m_combo_box_type_value)
     {
      //--- 현재 배열의 메모리 최적화
      OnClickButtonFree();
      //--- 다른 배열 유형으로 작업
      m_combo_box_type_value=(int)m_combo_box_type.Value();
      //--- 화면에 새 배열 유형을 표시
      if(m_combo_box_type_value==0)
         m_lbl_array_type.Text("배열 유형 = char");
      if(m_combo_box_type_value==1)
         m_lbl_array_type.Text("배열 유형 = int");
      if(m_combo_box_type_value==2)
         m_lbl_array_type.Text("배열 유형 = float");
      if(m_combo_box_type_value==3)
         m_lbl_array_type.Text("배열 유형 = double");
      if(m_combo_box_type_value==4)
         m_lbl_array_type.Text("배열 유형 = long");
     }
  }
//--- CMemoryControl 클래스 개체
CMemoryControl ExtDialog;
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 기능                                             |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 대화상자 생성
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- 시작
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 해제 기능                                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 차트 이벤트 기능                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayGetAsSeries

배열 인덱스의 방향을 확인.

```
bool  ArrayGetAsSeries(
   const void&  array[]    // 확인할 배열
   );
```

매개변수

배열

[in] 확인된 배열.

반환 값

지정된 배열에 AS_SERIES 플래그가 설정되면 true를 반환하며 즉, 배열에 대한 액세스가 시계열처럼 앞뒤로 수행됩니다. 시계열은 시계열 요소의 인덱싱을 처음부터 끝까지(최신 데이터부터 이전 데이터까지) 수행한다는 점에서 일반 배열과는 다릅니다.

참고

배열이 시계열에 속하는지 확인하려면 ArrayIsSeries() 기능을 사용합니다. OnCalculate() 함수에 입력 매개변수로 전달된 가격 데이터 배열은 시계열과 같은 인덱싱 방향을 가질 필요가 없습니다. 필요한 인덱싱 방향은 ArraySetAsSeries() 함수를 사용하여 설정할 수 있습니다.

예:

```
#property description "지표는 다음 사이의 차이에 대한 절대값을 계산합니다"
#property description "별도의 하위 창에 표시하는 시가 및 종가 또는 고가 및 저가"
#property description "히스토그램으로서."
//--- 지표 설정
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- 플롯
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- 매개변수 입력
input bool InpAsSeries=true; // 지표 버퍼의 인덱싱 방향
input bool InpPrices=true;   // 계산 가격 (true - 시가,종가; false - 고가,저가)
//--- 지표 버퍼
double ExtBuffer[];
//+------------------------------------------------------------------+
//| 지표 값 계산                                                     |
//+------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
  {
//--- 막대 계산을 위한 시작 변수
   int start=prev_calculated;
//--- 지표 값이 이전 틱에서 이미 계산된 경우 마지막 막대에서 작업
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated>0)
      start--;
//--- 배열의 인덱싱 방향 정의
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- 필요한 경우 인덱싱 방향을 직접 방향으로 교체
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- 지표 값 계산
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 바인드
   SetIndexBuffer(0,ExtBuffer);
//--- 지표 버퍼에 인덱싱 요소 설정
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- 지표가 얼마로 계산되는지 확인
   if(InpPrices)
     {
      //--- 시가 및 종가
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- 지표 색상 설정
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
      //--- 고가 및 저가
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- 지표 색상 설정
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 플래그 값에 따라 지표를 계산
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

추가 참조

시계열에 액세스, ArraySetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArrayInitialize

함수는 사전 설정된 값으로 숫자 배열을 초기화.

char 유형 배열 초기화하는 경우

```
int  ArrayInitialize(
   char     array[],      // 초기화된 배열
   char     value         // 설정될 값
   );
```

short 유형의 배열을 초기화 하는 경우

```
int  ArrayInitialize(
   short    array[],      // 초기화된 배열
   short    value         // 설정될 값
   );
```

int 유형의 배열을 초기화하는 경우

```
int  ArrayInitialize(
   int      array[],      // 초기화된 배열
   int      value         // 설정될 값
   );
```

long 유형의 배열을 초기화하는 경우

```
int  ArrayInitialize(
   long     array[],      // 초기화된 배열
   long     value         // 설정될 값
   );
```

float 유형의 배열을 초기화하는 경우

```
int  ArrayInitialize(
   float    array[],      // 초기화된 배열
   float    value         // 설정될 값
   );
```

double 유형의 배열을 초기화 하는 경우

```
int  ArrayInitialize(
   double   array[],      // 초기화된 배열
   double   value         // 설정될 값
   );
```

bool 유형의 배열을 초기화하는 경우

```
int  ArrayInitialize(
   bool     array[],      // 초기화된 배열
   bool     value         // 설정될 값
   );
```

© 2000-2025, MetaQuotes Ltd.

uint 유형의 배열을 초기화하는 경우

```
int  ArrayInitialize(
   uint    array[],     // 초기화된 배열
   uint    value        // 설정될 값
   );
```

매개변수

array[]

[out] 초기화해야하는 숫자 배열.

값

[in] 모든 배열 요소에 설정해야 하는 새 값.

반환 값

초기화된 요소의 수.

참고

ArrayResize() 함수를 사용하면 메모리를 물리적으로 재배치하지 않고도 추가 확장을 위해 예약된 배열의 크기를 설정할 수 있습니다. 메모리 재배치 작업이 상당히 느리기 때문에 더 나은 성능을 위해 구현됩니다.

ArrayInitialize(array, init_val)를 사용하여 배열을 초기화하는 것이 아니라 이 배열에 할당된 동일한 예약 요소 값으로 초기화하는 것을 의미하지 않습니다. *배열*을 추가로 확장할 때 ArrayResize() 함수를 사용하여 요소가 배열 끝에 추가되고 해당 값이 정의되지 않으며 대부분의 경우 *init_value*와 같지 않습니다.

예:

```
void OnStart()
  {
//--- 동적 배열
   double array[];
//--- 100개 요소의 배열 크기를 설정하고 다른 10개의 요소 버퍼를 예약합니다
   ArrayResize(array,100,10);
//--- EMPTY_VALUE=DBL_MAX 값을 사용하여 배열 요소를 초기화
   ArrayInitialize(array,EMPTY_VALUE);
   Print("초기화 후 마지막 요소 10개 값");
   for(int i=90;i<100;i++) printf("array[%d] = %G",i,array[i]);
//--- 배열을 5개 요소로 확장
   ArrayResize(array,105);
   Print("ArrayResize(array,105) 후 마지막 요소 10개 값");
//--- 예약 버퍼에서 5개의 마지막 요소 값을 가져왔습니다
   for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayFill

함수는 지정된 값으로 배열을 채움.

```
void  ArrayFill(
   void&  array[],      // 배열
   int    start,         // 시작 인덱스
   int    count,         // 채울 요소의 수
   void   value          // 값
   );
```

매개변수

array[]

[out] 단순 유형 배열 (char, uchar, short, ushort, int, uint, long, ulong, bool, color, datetime, float, double).

시작

[in] 시작 인덱스. 이 경우 지정된 AS_SERIES flag 플래그가 무시됩니다.

카운트

[in] 채울 요소의 수.

값

[in] 배열을 채울 값.

반환 값

반환 값이 없습니다.

참고

ArrayFill() 함수가 호출되면 항상 정상 인덱싱 방향(좌측에서 오른쪽으로)이 포함됩니다. 즉, ArraySetAsSeries() 함수를 사용하여 배열 요소에 액세스하는 순서 변경이 무시됩니다.

ArrayFill() 함수로 처리하면 다차원 배열이 1차원으로 표시됩니다. 예를 들어 array[2][4]는 array[8]로 처리됩니다. 따라서 이 배열에서 작업할 때 초기 요소의 인덱스를 5와 동일하게 지정할 수 있습니다. 따라서 array[2][4]에 대한 ArrayFill(array, 5, 2, 3.14) 호출은 array[1][1] 및 array[1][2] 요소를 3.14로 채웁니다.

예:

```
void OnStart()
  {
//--- 동적 배열을 선언
   int a[];
//--- 크기 설정
   ArrayResize(a,10);
//--- 첫 5개 요소를 123으로 채우기
   ArrayFill(a,0,5,123);
//--- 다음 5개 요소를 456로 채우기
   ArrayFill(a,5,5,456);
//--- 값 표시
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayIsDynamic

이 함수는 배열이 동적인지 여부를 확인.

```
bool  ArrayIsDynamic(
   const void&  array[]    // 확인된 배열
   );
```

매개변수

array[]

[in] 확인된 배열.

반환 값

선택한 배열이 동적인면 true를, 그렇지 않으면 false를 반환.

예:

```
#property description "이 지표는 값을 계산하지 않습니다. 단 한번 시도합니다"
#property description "ArrayFree() 기능을 호출은 3개의 배열(동적 배열 하나, 정적 배열 하나)0
#property description "지표 버퍼. 결과는 엑스퍼트 저널에 표시됩니다."
//--- 지표 설정
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- 글로벌 변수
double ExtDynamic[];   // 동적 배열
double ExtStatic[100]; // 정적 배열
bool   ExtFlag=true;   // 플래그
double ExtBuff[];      // 지표 버퍼
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 배열에 메모리 할당
   ArrayResize(ExtDynamic,100);
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 단일 분석을 실시
   if(ExtFlag)
     {
      //--- 배열에 여유 메모리 시도
      //--- 1. 동적 배열
      Print("+============================+");
      Print("1. 동적 배열 확인:");
      Print("메모리를 확보하기 전의 크기 = ",ArraySize(ExtDynamic));
      Print("동적 배열인지 확인 = ",ArrayIsDynamic(ExtDynamic) ? "예" : "아니오");
      //--- 배열 메모리 사용 시도
      ArrayFree(ExtDynamic);
      Print("메모리 확보한 후의 크기 = ",ArraySize(ExtDynamic));
      //--- 2. 정적 배열
      Print("2. 정적 배열 확인:");
      Print("메모리를 확보하기 전의 크기 = ",ArraySize(ExtStatic));
      Print("동적 배열인지 확인 = ",ArrayIsDynamic(ExtStatic) ? "예" : "아니오");
      //--- 배열 메모리 사용 시도
      ArrayFree(ExtStatic);
      Print("메모리를 확보한 후의 크기 = ",ArraySize(ExtStatic));
      //--- 3. 지표 버퍼
      Print("3. 지표 버퍼 확인:");
      Print("메모리 확보 전의 크기 = ",ArraySize(ExtBuff));
      Print("동적 배열인지 확인 = ",ArrayIsDynamic(ExtBuff) ? "예" : "아니오");
      //--- 배열 메모리 사용 시도
      ArrayFree(ExtBuff);
      Print("메모리 확보한 후의 크기 = ",ArraySize(ExtBuff));
      //--- 플래그 값 변경
      ExtFlag=false;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

추가 참조

시계열 및 지표에 대한 액세스

© 2000-2025, MetaQuotes Ltd.

ArrayIsSeries

이 함수는 배열이 시계열인지 여부를 확인.

```
bool  ArrayIsSeries(
   const void&  array[]    // 확인된 배열
   );
```

매개변수

array[]

[in] 확인된 배열.

반환 값

선택된 배열이 시계열 배열이면 true를 반환하고, 그렇지 않으면 false를 반환합니다. OnCalculate() 함수에 매개변수로 전달된 배열에서 ArrayGetAsSeries()를 통한 배열 요소에 액세스하는 순서를 확인해야 합니다.

예:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- 플롯 라벨1
#property indicator_label1  "라벨1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 지표 버퍼
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(ArrayIsSeries(open))
      Print("open[]이 시계열입니다");
   else
      Print("open[]이 시계열이 아닙니다!!!");
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

추가 참조

시계열 및 지표에 대한 액세스

© 2000-2025, MetaQuotes Ltd.

ArrayMaximum

다차원 숫자 배열의 첫 번째 차원에서 가장 큰 요소를 검색.

```
int  ArrayMaximum(
   const void&   array[],             // 검색할 배열
   int           start=0,             // 검사를 시작할 인덱스
   int           count=WHOLE_ARRAY    // 확인된 요소의 수
   );
```

매개변수

array[]

[in] 검색이 이루어지는 숫자 배열.

start=0

[in] 확인을 시작할 인덱스.

count=WHOLE_ARRAY

[in] 검색할 요소의 수. 기본적으로 전체 배열에서 검색 (count=WHOLE_ARRAY).

반환 값

함수는 배열 일련번호를 고려하여 발견된 요소의 인덱스를 반환합니다. 실패 시 -1을 반환.

참고

최대값을 검색하는 동안 AS_SERIES 플래그 값이 고려됩니다.

ArrayMaximum 및 ArrayMinimum 함수는 모든 차원 배열을 매개변수로 허용합니다. 그러나 검색은 항상 첫 번째 (0) 차원에 적용됩니다.

예:

```
#property description "지표에는 현재 지표에 더 큰 시간 프레임의 캔들스틱이 표시됩니다"
//--- 지표 설정
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- 플롯 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- 플롯 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- 플롯 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- 플롯 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
//---- 플롯 5
#property indicator_label5  "BullBody"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_color5  clrOlive,clrOlive
//---- 플롯 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- 플롯 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- 플롯 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- 사전정의된 상수
#define INDICATOR_EMPTY_VALUE 0.0
//--- 매개변수 입력
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // 지표 계산을 위한 시간 프레임
input datetime        InpDateStart=D'2013.01.01 00:00'; // 분석 시작 일자
//--- 캔들스틱에 대한 지표 버퍼
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- 강세 캔들스틱에 대한 지표 버퍼
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- 글로벌 변수
datetime ExtTimeBuff[];      // 더 큰 시간 프레임의 시간 버퍼
int      ExtSize=0;          // 시간 버퍼 크기
int      ExtCount=0;         // 인덱스 내부 시간 버퍼
int      ExtStartPos=0;      // 지표 계산을 위한 초기 포지션
bool     ExtStartFlag=true;  // 초기 포지션 수신을 위한 보조 플래그
datetime ExtCurrentTime[1];  // 더 큰 시간 프레임의 막대 생성의 마지막 시간
datetime ExtLastTime;        // 더 큰 시간 프레임에서 계산이 수행된 마지막 시간
bool     ExtBearFlag=true;   // 지표 버퍼에 데이터를 쓰는 순서를 정의하는 플래그
bool     ExtBullFlag=true;   // 강세 지표 버퍼에 데이터를 쓰는 순서를 정의하는 플래그
int      ExtIndexMax=0;      // 배열의 최대 ㅇ요소 인덱스
int      ExtIndexMin=0;      // 배열의 최소 요소 인덱스
int      ExtDirectionFlag=0; // 현재 캔들스틱에 대한 가격 이동 방향
//--- 정확한 드로잉을 위해 캔들스틱의 시가와 종가 사이를 바꿈
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 캔들스틱의 기본 부분 채우기                                      |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- 배열에서 최대 및 최소 요소의 인덱스를 찾기
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // 최대(고가)
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // 최소(저가)
//--- 현재 시간 프레임에서 몇 개의 막대를 채울 것인지를 정의
   int count=fill_index-start+1;
//--- 첫 막대의 종가가 마지막 막대의 종가보다 높으면 캔들스틱은 약세입니다
   if(open[start]>close[last])
     {
      //--- 그 전에 캔들스틱이 강세였다면, 강세 지표의 값을 지웁니다
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- 약세 캔들스틱
      ExtDirectionFlag=-1;
      //--- 캔들스틱 생성
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- 함수 나가기
      return;
     }
//--- 첫 번째 막대의 종가가 마지막 막대의 종가보다 낮으면 캔들스틱은 강세입니다
   if(open[start]<close[last])
     {
      //--- 그 전에 캔들스틱이 약세였다면, 약세 지표 버퍼 값을 지웁니다
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- 강세 캔들스틱
      ExtDirectionFlag=1;
      //--- 캔들스틱 생성
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- 함수 나가기
      return;
     }
//--- 만약 여러분이 기능의 이 부분에 있다면, 첫 번째 막대의 오픈 가격은 다음과 같습니다
//--- 마지막 막대의 종가; 그러한 캔들스틱은 약세로 간주됩니다
//--- 그 전에 촛대가 강세였다면, 강세 지표 버퍼의 값을 지웁니다
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- 약세 캔들스틱
   ExtDirectionFlag=-1;
//--- 종가 및 시가가 동일할 경우, 정확한 표시를 위해 시프트를 사용
   if(high[index_max]!=low[index_min])
```

© 2000-2025, MetaQuotes Ltd.

```
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| 캔들스틱의 끝 채우기                                             |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- 단 하나의 막대도 그리지 않습니다
   if(last-start==0)
      return;
//--- 첫 막대의 종가가 마지막 막대의 종가보다 높으면 캔들스틱은 약세입니다
   if(open[start]>close[last])
     {
      //--- 캔들스틱 끝 생성
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- 함수 나가기
      return;
     }
//--- 첫 번째 막대의 종가가 마지막 막대의 종가보다 낮으면 캔들스틱은 강세입니다
   if(open[start]<close[last])
     {
      //--- 캔들스틱 끝 생성
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- 함수 나가기
      return;
     }
//--- 만약 여러분이 기능의 이 부분에 있다면, 첫 번째 막대의 오픈 가격은 다음과 같습니다
//--- 마지막 막대의 종가; 그러한 캔들스틱은 약세로 간주됩니다
//--- 캔들스틱 끝 생성
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEr
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- 지표 기간 확인
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- 가격 데이터를 전면에 표시
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- 바인딩 지표 버퍼
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- 지표 생성을 위한 일부 속성 값 설정
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // 그래픽 구성 유형
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // 선 그리기 스타일
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // 선 그리기 굵기
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 계산된 막대가 아직 없는 경우
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated==0)
     {
      //--- 더 큰 시간 프레임의 도착 시간 수신
      if(!GetTimeData())
         return(0);
     }
//--- 직접 인덱싱 설정
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- 막대 계산을 위한 시작 변수
   int start=prev_calculated;
//--- 막대가 생성된 경우, 막대의 지표 값을 다시 계산
   if(start!=0 && start==rates_total)
      start--;
//--- 지표 값 계산을 위한 루프
   for(int i=start;i<rates_total;i++)
     {
      //--- 빈 값으로 지표 버퍼의 i 요소 채우기
      FillIndicatorBuffers(i);
      //--- InpDateStart 날짜부터 막대 계산 수행
      if(time[i]>=InpDateStart)
        {
         //--- 처음으로 값을 표시할 포지션 정의
         if(ExtStartFlag)
           {
            //--- 초기 막대의 수를 저장
            ExtStartPos=i;
            //--- 더 긴 시간[i]을 초과한 첫 번째 날짜를 정의
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- 이 블록에 다시 들어가지 않도록 플래그 값을 변경
            ExtStartFlag=false;
           }
         //--- 배열에 아직 요소가 있는지 확인
         if(ExtCount<ExtSize)
           {
            //--- 현재 시간 프레임의 값이 더 큰 시간 프레임의 값에 도달할 때까지 기다림
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- 캔들스틱의 주요 부분을 그림 (마지막 막대와 끝에서 2번째 막대 사이의 영역을
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Ext
               //--- 캔들스틱의 끝 부분을 채움 (마지막 막대와 끝에서 2번째 막대 사이의 영역)
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
               //--- 다음 캔들스틱을 그리기 위한 초기 포지션을 옮김
               ExtStartPos=i;
```

© 2000-2025, MetaQuotes Ltd.

```
                //--- 배열 카운터 증가
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- 배열 값 재설정
             ResetLastError();
             //--- 더 큰 시간 프레임에서 마지막 날짜 수신
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Data copy error, code = ",GetLastError());
                return(0);
               }
             //--- 새 날짜가 이후인 경우 캔들스틱 생성을 중지
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- 주 지표 버퍼에서 마지막 막대외 끝에서 2번째 막대 사이의 영역을 지움
                ClearEndOfBodyMain(i-1);
                //--- 보조 지표 버퍼를 사용하여 해당 구역에 기입
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- 다음 캔들스틱을 그리기 위한 초기 포지션을 옮김
                ExtStartPos=i;
                //--- 가격 방향 플래그 재설정
                ExtDirectionFlag=0;
                //--- 새 마지막 날짜를 저장
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- 캔들스틱 생성
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지정된 지표 기간의 정확성을 확인                                 |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- 지표 기간은 표시된 시간 프레임을 초과해야 합니다
   if(current_period>=high_period)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("오류! 지표 기간의 값이 현지 시간 프레임의 값을 초과해야 합니다!");
      return(false);
     }
//--- 지표 기간이 1주일 또는 1개월이면 해당 기간이 올바릅니다
   if(high_period>32768)
      return(true);
//--- 기간 값을 분으로 변환
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- 지표 기간은 표시된 시간 프레임의 배수가 되어야 합니다
   if(high_period%current_period!=0)
     {
      Print("오류! 지표 기간의 값은 현재 시간 프레임 값의 배수가 되어야 합니다!");
      return(false);
     }
//--- 지표 기간는 표시된 시간 프레임을 3배 이상 초과해야 합니다
   if(high_period/current_period<3)
     {
      Print("오류! 지표 기간은 현재 기간의 3배 이상 초과해야 합니다!");
      return(false);
     }
//--- 지표 기간이 현재 시간 프레임에 대해 올바릅니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 더 큰 시간 프레임에서 시간 데이터 수신                           |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 현재 시간에 대한 모든 데이터를 복사
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- 오류 코드 수신
      int code=GetLastError();
      //--- 에러 메시지 출력
      PrintFormat("Data copy error! %s",code==4401
                  ? "내역이 아직 업데이트되고 있습니다!"
                  : "Code = "+IntegerToString(code));
      //--- false를 반환하여 데이터 다운로드를 반복해서 시도
      return(false);
     }
//--- 배열 크기 수신
   ExtSize=ArraySize(ExtTimeBuff);
//--- 배열의 루프 인덱스를 0으로 설정
   ExtCount=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 시간 프레임에서 현재 캔들스틱의 포지션을 0으로 설정
   ExtStartPos=0;
   ExtStartFlag=true;
//--- 더 큰 시간 프레임의 마지ㅁㄱ 시간 값을 저장
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------------------+
//| 기능은 캔들스틱의 주요 부분을 구성합니다. 플래그에 따라 |
//| 값, 함수는 데이터 및 배열을 정의합니다                          |
//| 정확하게 표시하기 위해 사용하기 위해.                                          |
//+------------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- 플래그의 값 확인
   if(flag)
     {
      //--- 캔들스틱의 몸통 생성
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- 캔들스틱의 그림자 생성
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- 캔들스틱의 몸통 생성
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- 캔들스틱의 그림자 생성
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------+
//| 기능은 캔들스틱의 끝을 구성합니다. 플래그의 값에 따라, |
//| 함수는 데이터 및 배열을 정의합니다                                  |
//| 정확하게 표시하기 위해 사용하기 위해.                                          |
//+------------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- 플래그의 값 확인
   if(flag)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 캔들스틱의 몸통 끝을 생성
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- 캔들스틱의 그림자 끝을 생성
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- 플래그의 값을 반대로 변경
      flag=false;
     }
   else
     {
      //--- 캔들스틱의 몸통 끝을 생성
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- 캔들스틱의 그림자 끝을 생성
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- 플래그의 값을 반대로 변경
      flag=true;
     }
  }
//+------------------------------------------------------------------------------+
//| 캔들스틱의 끝(마지막과 끝에서 2번째 사이의 영역)을 지웁니다 |
//| bar)                                                                         |
//+------------------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------------------+
//| 캔들스틱 지우기                                                              |
//+------------------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- 확인
   if(count!=0)
     {
      //--- 빈 값으로 지표 버퍼 채우기
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------------------+
//| 캔들스틱의 주요 부분 생성                                                    |
//+------------------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
  {
//--- 확인
```

© 2000-2025, MetaQuotes Ltd.

```
   if(count!=0)
     {
      //--- 값으로 지표 버퍼 채우기
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------------------+
//| 캔들스틱의 끝 생성                                                           |
//+------------------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- 값으로 지표 버퍼 채우기
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| 지표 버퍼의 i 요소를 빈 값으로 채우기          |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- 지표 버퍼의 셀에 빈 값을 설정
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayMinimum

다차원 숫자 배열의 첫 번째 차원에서 가장 낮은 요소를 검색.

```
int  ArrayMinimum(
   const void&   array[],              // 검색할 배열
   int           start=0,              // 검사를 시작할 인덱스
   int           count=WHOLE_ARRAY     // 확인된 요소의 수
   );
```

매개변수

array[]

[in] 검색이 이루어지는 숫자 배열.

start=0

[in] 확인을 시작할 인덱스.

count=WHOLE_ARRAY

[in] 검색할 요소의 수. 기본적으로 전체 배열에서 검색 (count=WHOLE_ARRAY).

반환 값

함수는 배열 일련번호를 고려하여 발견된 요소의 인덱스를 반환합니다. 실패 시 -1을 반환.

참고

최소값을 검색하는 동안 AS_SERIES 플래그 값이 고려됩니다.

ArrayMaximum 및 ArrayMinimum 함수는 모든 차원 배열을 매개변수로 허용합니다. 그러나 검색은 항상 첫 번째 (0) 차원에 적용됩니다.

예:

```
#property description "지표에는 현재 지표에 더 큰 시간 프레임의 캔들스틱이 표시됩니다"
//--- 지표 설정
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots   8
//---- 플롯 1
#property indicator_label1  "BearBody"
#property indicator_color1  clrSeaGreen,clrSeaGreen
//---- 플롯 2
#property indicator_label2  "BearBodyEnd"
#property indicator_color2  clrSeaGreen,clrSeaGreen
//---- 플롯 3
#property indicator_label3  "BearShadow"
#property indicator_color3  clrSalmon,clrSalmon
//---- 플롯 4
#property indicator_label4  "BearShadowEnd"
#property indicator_color4  clrSalmon,clrSalmon
//---- 플롯 5
#property indicator_label5  "BullBody"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_color5  clrOlive,clrOlive
//---- 플롯 6
#property indicator_label6  "BullBodyEnd"
#property indicator_color6  clrOlive,clrOlive
//---- 플롯 7
#property indicator_label7  "BullShadow"
#property indicator_color7  clrSkyBlue,clrSkyBlue
//---- 플롯 8
#property indicator_label8  "BullShadowEnd"
#property indicator_color8  clrSkyBlue,clrSkyBlue
//--- 사전정의된 상수
#define INDICATOR_EMPTY_VALUE 0.0
//--- 매개변수 입력
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // 지표 계산을 위한 시간 프레임
input datetime        InpDateStart=D'2013.01.01 00:00'; // 분석 시작 일자
//--- 캔들스틱에 대한 지표 버퍼
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- 강세 캔들스틱에 대한 지표 버퍼
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- 글로벌 변수
datetime ExtTimeBuff[];      // 더 큰 시간 프레임의 시간 버퍼
int      ExtSize=0;          // 시간 버퍼 크기
int      ExtCount=0;         // 인덱스 내부 시간 버퍼
int      ExtStartPos=0;      // 지표 계산을 위한 초기 포지션
bool     ExtStartFlag=true;  // 초기 포지션 수신을 위한 보조 플래그
datetime ExtCurrentTime[1];  // 더 큰 시간 프레임의 막대 생성의 마지막 시간
datetime ExtLastTime;        // 더 큰 시간 프레임에서 계산이 수행된 마지막 시간
bool     ExtBearFlag=true;   // 지표 버퍼에 데이터를 쓰는 순서를 정의하는 플래그
bool     ExtBullFlag=true;   // 강세 지표 버퍼에 데이터를 쓰는 순서를 정의하는 플래그
int      ExtIndexMax=0;      // 배열의 최대 ㅇ요소 인덱스
int      ExtIndexMin=0;      // 배열의 최소 요소 인덱스
int      ExtDirectionFlag=0; // 현재 캔들스틱에 대한 가격 이동 방향
//--- 정확한 드로잉을 위해 캔들스틱의 시가와 종가 사이를 바꿈
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 캔들스틱의 기본 부분 채우기                                      |
//+------------------------------------------------------------------+
void FillCandleMain(const double &open[],const double &close[],
                    const double &high[],const double &low[],
                    const int start,const int last,const int fill_index,
                    int &index_max,int &index_min)
  {
//--- 배열에서 최대 및 최소 요소의 인덱스를 찾기
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // 최대(고가)
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // 최소(저가)
//--- 현재 시간 프레임에서 몇 개의 막대를 채울 것인지를 정의
   int count=fill_index-start+1;
//--- 첫 막대의 종가가 마지막 막대의 종가보다 높으면 캔들스틱은 약세입니다
   if(open[start]>close[last])
     {
      //--- 그 전에 캔들스틱이 강세였다면, 강세 지표의 값을 지웁니다
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- 약세 캔들스틱
      ExtDirectionFlag=-1;
      //--- 캔들스틱 생성
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- 함수 나가기
      return;
     }
//--- 첫 번째 막대의 종가가 마지막 막대의 종가보다 낮으면 캔들스틱은 강세입니다
   if(open[start]<close[last])
     {
      //--- 그 전에 캔들스틱이 약세였다면, 약세 지표 버퍼 값을 지웁니다
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- 강세 캔들스틱
      ExtDirectionFlag=1;
      //--- 캔들스틱 생성
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- 함수 나가기
      return;
     }
//--- 만약 여러분이 기능의 이 부분에 있다면, 첫 번째 막대의 오픈 가격은 다음과 같습니다
//--- 마지막 막대의 종가; 그러한 캔들스틱은 약세로 간주됩니다
//--- 그 전에 촛대가 강세였다면, 강세 지표 버퍼의 값을 지웁니다
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- 약세 캔들스틱
   ExtDirectionFlag=-1;
//--- 종가 및 시가가 동일할 경우, 정확한 표시를 위해 시프트를 사용
   if(high[index_max]!=low[index_min])
```

© 2000-2025, MetaQuotes Ltd.

```
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start
   else
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     open[start],open[start]-ExtEmptyBodySize,high[index_max],
                     high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);
  }
//+------------------------------------------------------------------+
//| 캔들스틱의 끝 채우기                                             |
//+------------------------------------------------------------------+
void FillCandleEnd(const double &open[],const double &close[],
                   const double &high[],const double &low[],
                   const int start,const int last,const int fill_index,
                   const int index_max,const int index_min)
  {
//--- 단 하나의 막대도 그리지 않습니다
   if(last-start==0)
      return;
//--- 첫 막대의 종가가 마지막 막대의 종가보다 높으면 캔들스틱은 약세입니다
   if(open[start]>close[last])
     {
      //--- 캔들스틱 끝 생성
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- 함수 나가기
      return;
     }
//--- 첫 번째 막대의 종가가 마지막 막대의 종가보다 낮으면 캔들스틱은 강세입니다
   if(open[start]<close[last])
     {
      //--- 캔들스틱 끝 생성
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- 함수 나가기
      return;
     }
//--- 만약 여러분이 기능의 이 부분에 있다면, 첫 번째 막대의 오픈 가격은 다음과 같습니다
//--- 마지막 막대의 종가; 그러한 캔들스틱은 약세로 간주됩니다
//--- 캔들스틱 끝 생성
   if(high[index_max]!=low[index_min])
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
   else
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEr
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- 지표 기간 확인
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- 가격 데이터를 전면에 표시
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- 바인딩 지표 버퍼
   SetIndexBuffer(0,ExtBearBodyFirst);
   SetIndexBuffer(1,ExtBearBodySecond);
   SetIndexBuffer(2,ExtBearBodyEndFirst);
   SetIndexBuffer(3,ExtBearBodyEndSecond);
   SetIndexBuffer(4,ExtBearShadowFirst);
   SetIndexBuffer(5,ExtBearShadowSecond);
   SetIndexBuffer(6,ExtBearShadowEndFirst);
   SetIndexBuffer(7,ExtBearShadowEndSecond);
   SetIndexBuffer(8,ExtBullBodyFirst);
   SetIndexBuffer(9,ExtBullBodySecond);
   SetIndexBuffer(10,ExtBullBodyEndFirst);
   SetIndexBuffer(11,ExtBullBodyEndSecond);
   SetIndexBuffer(12,ExtBullShadowFirst);
   SetIndexBuffer(13,ExtBullShadowSecond);
   SetIndexBuffer(14,ExtBullShadowEndFirst);
   SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- 지표 생성을 위한 일부 속성 값 설정
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // 그래픽 구성 유형
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // 선 그리기 스타일
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // 선 그리기 굵기
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 계산된 막대가 아직 없는 경우
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated==0)
     {
      //--- 더 큰 시간 프레임의 도착 시간 수신
      if(!GetTimeData())
         return(0);
     }
//--- 직접 인덱싱 설정
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- 막대 계산을 위한 시작 변수
   int start=prev_calculated;
//--- 막대가 생성된 경우, 막대의 지표 값을 다시 계산
   if(start!=0 && start==rates_total)
      start--;
//--- 지표 값 계산을 위한 루프
   for(int i=start;i<rates_total;i++)
     {
      //--- 빈 값으로 지표 버퍼의 i 요소 채우기
      FillIndicatorBuffers(i);
      //--- InpDateStart 날짜부터 막대 계산 수행
      if(time[i]>=InpDateStart)
        {
         //--- 처음으로 값을 표시할 포지션 정의
         if(ExtStartFlag)
           {
            //--- 초기 막대의 수를 저장
            ExtStartPos=i;
            //--- 더 긴 시간[i]을 초과한 첫 번째 날짜를 정의
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- 이 블록에 다시 들어가지 않도록 플래그 값을 변경
            ExtStartFlag=false;
           }
         //--- 배열에 아직 요소가 있는지 확인
         if(ExtCount<ExtSize)
           {
            //--- 현재 시간 프레임의 값이 더 큰 시간 프레임의 값에 도달할 때까지 기다림
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- 캔들스틱의 주요 부분을 그림 (마지막 막대와 끝에서 2번째 막대 사이의 영역을
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,Ext
               //--- 캔들스틱의 끝 부분을 채움 (마지막 막대와 끝에서 2번째 막대 사이의 영역)
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
               //--- 다음 캔들스틱을 그리기 위한 초기 포지션을 옮김
               ExtStartPos=i;
```

© 2000-2025, MetaQuotes Ltd.

```
                //--- 배열 카운터 증가
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- 배열 값 재설정
             ResetLastError();
             //--- 더 큰 시간 프레임에서 마지막 날짜 수신
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Data copy error, code = ",GetLastError());
                return(0);
               }
             //--- 새 날짜가 이후인 경우 캔들스틱 생성을 중지
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- 주 지표 버퍼에서 마지막 막대외 끝에서 2번째 막대 사이의 영역을 지움
                ClearEndOfBodyMain(i-1);
                //--- 보조 지표 버퍼를 사용하여 해당 구역에 기입
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- 다음 캔들스틱을 그리기 위한 초기 포지션을 옮김
                ExtStartPos=i;
                //--- 가격 방향 플래그 재설정
                ExtDirectionFlag=0;
                //--- 새 마지막 날짜를 저장
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- 캔들스틱 생성
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지정된 지표 기간의 정확성을 확인                                 |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- 지표 기간은 표시된 시간 프레임을 초과해야 합니다
   if(current_period>=high_period)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("오류! 지표 기간의 값이 현지 시간 프레임의 값을 초과해야 합니다!");
      return(false);
     }
//--- 지표 기간이 1주일 또는 1개월이면 해당 기간이 올바릅니다
   if(high_period>32768)
      return(true);
//--- 기간 값을 분으로 변환
   if(high_period>30)
      high_period=(high_period-16384)*60;
   if(current_period>30)
      current_period=(current_period-16384)*60;
//--- 지표 기간은 표시된 시간 프레임의 배수가 되어야 합니다
   if(high_period%current_period!=0)
     {
      Print("오류! 지표 기간의 값은 현재 시간 프레임 값의 배수가 되어야 합니다!");
      return(false);
     }
//--- 지표 기간는 표시된 시간 프레임을 3배 이상 초과해야 합니다
   if(high_period/current_period<3)
     {
      Print("오류! 지표 기간은 현재 기간의 3배 이상 초과해야 합니다!");
      return(false);
     }
//--- 지표 기간이 현재 시간 프레임에 대해 올바릅니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 더 큰 시간 프레임에서 시간 데이터 수신                           |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- 오류 값 재설정
   ResetLastError();
//--- 현재 시간에 대한 모든 데이터를 복사
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
      //--- 오류 코드 수신
      int code=GetLastError();
      //--- 에러 메시지 출력
      PrintFormat("Data copy error! %s",code==4401
                  ? "내역이 아직 업데이트되고 있습니다!"
                  : "Code = "+IntegerToString(code));
      //--- false를 반환하여 데이터 다운로드를 반복해서 시도
      return(false);
     }
//--- 배열 크기 수신
   ExtSize=ArraySize(ExtTimeBuff);
//--- 배열의 루프 인덱스를 0으로 설정
   ExtCount=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 시간 프레임에서 현재 캔들스틱의 포지션을 0으로 설정
   ExtStartPos=0;
   ExtStartFlag=true;
//--- 더 큰 시간 프레임의 마지ㅁㄱ 시간 값을 저장
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------------------+
//| 기능은 캔들스틱의 주요 부분을 구성합니다. 플래그에 따라 |
//| 값, 함수는 데이터 및 배열을 정의합니다                          |
//| 정확하게 표시하기 위해 사용하기 위해.                                          |
//+------------------------------------------------------------------------------+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
  {
//--- 플래그의 값 확인
   if(flag)
     {
      //--- 캔들스틱의 몸통 생성
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- 캔들스틱의 그림자 생성
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- 캔들스틱의 몸통 생성
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- 캔들스틱의 그림자 생성
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------+
//| 기능은 캔들스틱의 끝을 구성합니다. 플래그의 값에 따라,  |
//| 함수는 데이터 및 배열을 정의합니다                                  |
//| 정확하게 표시하기 위해 사용하기 위해.                                          |
//+------------------------------------------------------------------------------+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
  {
//--- 플래그의 값 확인
   if(flag)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 캔들스틱의 몸통 끝을 생성
      FormEnd(body_fst,body_snd,fst_value,snd_value,end);
      //--- 캔들스틱의 그림자 끝을 생성
      FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
      //--- 플래그의 값을 반대로 변경
      flag=false;
     }
   else
     {
      //--- 캔들스틱의 몸통 끝을 생성
      FormEnd(body_fst,body_snd,snd_value,fst_value,end);
      //--- 캔들스틱의 그림자 끝을 생성
      FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
      //--- 플래그의 값을 반대로 변경
      flag=true;
     }
  }
//+------------------------------------------------------------------------------------
//| 캔들스틱의 끝(마지막과 끝에서 2번째 사이의 영역)을 지웁니다      |
//| 막대)
//+------------------------------------------------------------------------------------
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------+
//| 캔들스틱 비우기                                                  |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- 확인
   if(count!=0)
     {
      //--- 빈 값으로 지표 버퍼 채우기
      ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
      ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
     }
  }
//+------------------------------------------------------------------+
//| 캔들스틱의 주요 부분 생성                          |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
  {
//--- 확인
```

© 2000-2025, MetaQuotes Ltd.

```
   if(count!=0)
     {
      //--- 값으로 지표 버퍼 채우기
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| 캔들스틱의 끝 생성                                               |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- 값으로 지표 버퍼 채우기
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| 지표 버퍼의 i 요소를 빈 값으로 채우기                            |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- 지표 버퍼의 셀에 빈 값을 설정
   ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
   ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayPrint

단순 유형 또는 단순 구조의 배열을 저널로 인쇄.

```
void  ArrayPrint(
   const void&   array[],              // 인쇄된 배열
   uint          digits=_Digits,       // 소수 자릿수
   const string  separator=NULL,       // 구조 필드 값의 분리자
   ulong         start=0,              // 첫 번째 인쇄 요소 인덱스
   ulong         count=WHOLE_ARRAY,    // 인쇄된 요소의 수
   ulong         flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_
   );
```

매개변수

array[]

[in] 단순 유형 또는 단순 구조의 배열.

digits=_Digits

[in] 실수 유형의 소수 자릿수. 기본값은 _Digits입니다.

separator=NULL

[in] 구조 요소 필드 값의 분리자. 기본값 NULL은 빈 줄을 의미합니다. 이 경우 공백이 분리자로 사용됩니다.

start=0

[in] 첫 번째 인쇄된 배열 요소의 인덱스. 기본적으로 영점 인덱스에서 인쇄됩니다.

count=WHOLE_ARRAY

[in] 인쇄할 배열의 요소 수. 기본적으로 전체 배열이 표시됩니다 (count=WHOLE_ARRAY).

flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_ALIGN

[in] 플래그 조합이 출력 모드를 설정 기본적으로 모든 플래그가 활성화됩니다:

- ARRAYPRINT_HEADER – 구조 배열에 대한 헤더 인쇄
- ARRAYPRINT_INDEX – 인덱스를 좌측에 인쇄
- ARRAYPRINT_LIMIT – 첫 100개 및 마지막 100개 배열 요소만 인쇄. 큰 배열의 일부만 인쇄하려는 경우에 사용.
- ARRAYPRINT_ALIGN – 인쇄된 값의 정렬을 활성화하며, 숫자는 우측에 정렬되고 선은 좌측에 정렬됩니다.
- ARRAYPRINT_DATE – 일시를 인쇄할 때 dd.mm.yyyy 서식에서 날짜를 인쇄합니다
- ARRAYPRINT_MINUTES – 일시를 인쇄할 때 HH:MM 서식에서 시간을 인쇄합니다
- ARRAYPRINT_SECONDS – 일시를 인쇄할 때 HH:MM:SS 서식에서 시간을 인쇄합니다

반환 값

아니오

참고

ArrayPrint()는 일부 구조 배열 필드를 저널로 인쇄하지 않으며, 배열 및 개체 포인터 필드는 건너뜁니다. 이 열들은 더 편리한 프리젠테이션을 위해 인쇄되는 것이 아닙니다. 모든 구조 필드를 인쇄해야 하는 경우 원하는 형식의 대략 인쇄 기능을 직접 작성해야 합니다.

© 2000-2025, MetaQuotes Ltd.

예:

```
//--- 마지막 막대 10개 값을 인쇄
   MqlRates rates[];
   if(CopyRates(_Symbol,_Period,1,10,rates))
     {
      ArrayPrint(rates);
      Print("Check\n[time]\t[open]\t[high]\t[low]\t[close]\t[tick_volume]\t[spread]\t
      for(int i=0;i<10;i++)
        {
         PrintFormat("[%d]\t%s\t%G\t%G\t%G\t%G\t%G\t%G\t%I64d\t",i,
         TimeToString(rates[i].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS),
         rates[i].open,rates[i].high,rates[i].low,rates[i].close,
         rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
   else
      PrintFormat("CopyRates failed, error code=%d",GetLastError());
//--- 인쇄의 예
/*
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295         18110       10   17
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.11930 1.12747         17829        9   15
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744         13458       10    9
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194         15362        9   12
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172         16833        9   12
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052         15933        8   10
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528         11888        9    8
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915          7284       10    5
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904          8710        9    6
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263          8956        7    7
  확인
   [time] [open] [high] [low] [close] [tick_volume] [spread] [real_volume]
   [0] 2016.11.09 04:00:00 1.11242 1.12314 1.11187 1.12295 18110 10 17300175000
   [1] 2016.11.09 05:00:00 1.12296 1.12825 1.1193 1.12747 17829 9 15632176000
   [2] 2016.11.09 06:00:00 1.12747 1.12991 1.12586 1.12744 13458 10 9593492000
   [3] 2016.11.09 07:00:00 1.12743 1.12763 1.11988 1.12194 15362 9 12352245000
   [4] 2016.11.09 08:00:00 1.12194 1.12262 1.11058 1.11172 16833 9 12961333000
   [5] 2016.11.09 09:00:00 1.11173 1.11348 1.10803 1.11052 15933 8 10720384000
   [6] 2016.11.09 10:00:00 1.11052 1.11065 1.10289 1.10528 11888 9 8084811000
   [7] 2016.11.09 11:00:00 1.10512 1.11041 1.10472 1.10915 7284 10 5087113000
   [8] 2016.11.09 12:00:00 1.10915 1.11079 1.10892 1.10904 8710 9 6769629000
   [9] 2016.11.09 13:00:00 1.10904 1.10913 1.10223 1.10263 8956 7 7192138000
*/
```

추가 참조

FileSave, FileLoad

© 2000-2025, MetaQuotes Ltd.

ArrayRange

함수는 선택한 배열 차원에 있는 요소의 수를 반환

```
int  ArrayRange(
   const void&   array[],      // 확인할 배열
   int           rank_index    // 차원 인덱스
   );
```

매개변수

array[]

[in] 확인된 배열.

rank_index

[in] 차원 인덱스.

반환 값

선택된 배열 차원의 요소 수

참고

인덱스가 0에서 시작되므로 배열 차원의 수가 마지막 차원의 인덱스보다 하나 큽니다.

예:

```
void OnStart()
  {
//--- 4차원 배열을 생성
   double array[][5][2][4];
//--- 0차원의 크기를 설정
   ArrayResize(array,10,10);
//--- 차원 인쇄
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- i 차원의 크기만큼 수신
      temp=ArrayRange(array,i);
      //--- 인쇄
      PrintFormat("dim = %d, range = %d",i,temp);
     }
//--- 결과
// dim = 0, range = 10
// dim = 1, range = 5
// dim = 2, range = 2
// dim = 3, range = 4
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrayResize

함수는 첫 번째 차원에 대해 새 크기를 설정합니다

```
int  ArrayResize(
   void&  array[],              // 참조로 전달된 배열
   int    new_size,             // 새 배열 크기
   int    reserve_size=0        // 예비 크기 값(초과값)
   );
```

매개변수

array[]

[out] 크기를 변경할 수 있는 배열.

new_size

[in] 첫 번째 차원의 새 크기.

reserve_size=0

[in] 예비로 가져올 분산 크기.

반환 값

성공적으로 실행되면 크기 조정 후 배열에 포함된 모든 요소의 수가 반환되고, 그렇지 않으면 -1이 반환되며 배열의 크기가 조정되지 않습니다.

ArrayResize()를 정적 배열, 시계열 또는 지표 버퍼에 적용하면 배열 크가가 동일하게 유지되고 이러한 배열은 재할당되지 않습니다. 이 경우, *new_size*<=ArraySize(*array*)이면, 함수는 *new_size*만 반환하며, 그렇지 않으면 -1 값을 반환됩니다.

참고

이 기능은 동적 배열에만 적용할 수 있습니다. SetIndexBuffer() 기능을 통해 지표 버퍼로 할당된 동적 배열의 크기는 변경할 수 없습니다. 지표 버퍼의 경우 크기 조정 작업은 모두 터미널의 런타임 하위 시스템에 의해 수행됩니다.

배열의 총 요소의 양은 2147483647을 초과할 수 없습니다.

메모리를 자주 할당할 경우 물리적 메모리 할당 수를 줄이려면 예약을 설정하는 세 번 째 변수를 사용하는 것이 좋습니다. 이후의 모든 ArrayResize 호출은 메모리를 물리적으로 재할당하지 않고 예약된 메모리 내에서 첫 번째 배열 차원의 크기만 변경합니다. 세 번째 매개변수는 물리적 메모리를 할당하는 동안에만 사용됩니다. 예를 들어:

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

이 경우, 메모리가 두 번 재할당되며, 첫 번째는 3000회 반복 루프(배열의 크기가 1000으로 설정됨)에 들어가기 전에 i를 사용하는 두 번째는 2000과 같습니다. 세 번째 매개변수를 건너뛰면 2000 개의 물리적 메모리 재할당이 발행하여 프로그램 속도가 느려집니다.

예:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 카운터
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- 빠른 버전의 데모를 위한 배열
   double arr[];
   ArrayResize(arr,100000,100000);
//--- 메모리 예약이 있는 모델이 얼마나 빨리 작동하는지 확인
   Print("--- 빠른 테스트: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
      //--- 100,000 요소의 예약을 지정하는 새 배열 크기를 설정합니다!
      ArrayResize(arr,i,100000);
      //--- 라운드 수에 도달하면 배열 크기와 소요 시간을 표시
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-sta
         start=now;
        }
     }
//--- 이제 메모리 예약이 없는 버전이 얼마나 느린지 보여줍니다
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- 느린 테스트: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- 추가 예약 없이 새 배열 크기를 설정
      ArrayResize(slow,i);
      //--- 라운드 수에 도달하면 배열 크기와 소요 시간을 표시
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-st
         start=now;
        }
     }
  }
//--- 스크립트의 샘플 결과
```

© 2000-2025, MetaQuotes Ltd.

```
/*
   Test_ArrayResize (EURUSD,H1)   --- Test Fast: ArrayResize(arr,100000,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(arr)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(arr)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(arr)=300000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   ---- Test Slow: ArrayResize(slow,100000)
   Test_ArrayResize (EURUSD,H1)   1. ArraySize(slow)=100000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   2. ArraySize(slow)=200000 Time=0 ms
   Test_ArrayResize (EURUSD,H1)   3. ArraySize(slow)=300000 Time=228511 ms
*/
```

추가 참조

ArrayInitialize

© 2000-2025, MetaQuotes Ltd.

ArrayInsert

원본 배열에서 지정된 수의 요소를 지정된 인덱스에서 시작하여 수신되는 요소에 삽입.

```
bool  ArrayInsert(
   void&        dst_array[],          // 배열 수신
   const void&  src_array[],          // 원본 배열
   uint         dst_start,            // 삽입할 수신자 배열 인덱스
   uint         src_start=0,          // 복사할 원본 배열 인덱스
   uint         count=WHOLE_ARRAY     // 삽입할 요소의 수
   );
```

매개변수

dst_array[]

[in][out] 수신 배열에 요소를 추가해야 합니다.

src_array[]

[in] 요소를 추가할 원본 배열.

dst_start

[in] 원본 배열에서 요소를 삽입하기 위한 수신 배열 인덱스.

src_start=0

[in] 원본 배열 요소를 삽입할 때 시작하는 수신 배열의 인덱스.

카운트

[in] 원본 배열에서 추가할 요소의 수 WHOLE_ARRAY는 지정된 인덱스에서 배열의 끝까지 모든 요소를 의미합니다.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류에 대한 정보를 가져오려면 GetLastError() 기능을 호출. 가능한 오류:

- 5052 - ERR_SMALL_ARRAY (*시작* 또는 *카운트* 매개변수가 잘못 설정되었거나 *src_array[]* 원본 배열이 비어있습니다),
- 5056 - ERR_SERIES_ARRAY (배열을 변경할 수 없습니다, 지표 버퍼),
- 4006 - ERR_INVALID_ARRAY (자체에 대한 복사가 허용되지 않거나, 배열 유형이 다르거나 클래스 유형이 다르거나, 클래스 개체 또는 소멸자 구조를 포함하는 고정 크기 배열이 있습니다),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (배열에 POD 구조가 포함되어 있지 않으므로 단순 복사가 불가능합니다),
- *dst_array[]* 수신 배열 크기를 변경하는 동안 ArrayRemove() 기능 설명에 오류가 발생했습니다.

참고

함수가 고정 크기 배열에 사용되는 경우 *dst_array[]* 수신 배열의 크기는 변경되지 않습니다. *dst_start* 포지션에서 시작하여 수신 배열 요소가 우측 ("꺼짐" 요소의 마지막 *카운트*)으로 이동되고, 원본 배열에서 복사된 요소가 그 자리를 대신합니다.

SetIndexBuffer() 기능으로 지표 버퍼로 지정된 동적 배열에는 요소를 삽입할 수 없습니다. 지표 버퍼의 경우 터미널의 실행 하위 시스템에 의해 모든 크키 변경 작업이 수행됩니다.

© 2000-2025, MetaQuotes Ltd.

원본 배열에서 요소는 *src_start* 인덱스에서 시작하여 복삽됩니다. 원본 배열 크기는 변경되지 않습니다. 수신 배열에 추가할 요소가 원본 배열 요소에 대한 링크가 아닙니다. 즉, 두 배열의 이후 요소 변경 사항은 두 번째 배열에 반영되지 않습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 고정 크기 배열을 선언하고 값을 채우기
   int array_dest[10];
   for(int i=0;i<10;i++)
     {
      array_dest[i]=i;
     }
   //--- 원본 배열
   int array_source[10];
   for(int i=0;i<10;i++)
     {
      array_source[i]=10+i;
     }
//--- 요소를 삽입하기 전에 배열을 표시
   Print("ArrayInsert() 호출 전");
   ArrayPrint(array_dest);
   ArrayPrint(array_source);
//--- 원본 배열에서 요소 3개를 삽입하고 수신 배열의 새 집합을 표시
   ArrayInsert(array_dest,array_source,4,0,3);
   Print("ArrayInsert() 호출 후 ");
   ArrayPrint(array_dest);
/*
  실행 결과
   Before calling ArrayInsert()
   0 1 2 3 4 5 6 7 8 9
   After calling ArrayInsert()
   0 1 2 3 10 11 12 7 8 9
*/
```

추가 참조

ArrayRemove, ArrayCopy, ArrayResize, ArrayFree

© 2000-2025, MetaQuotes Ltd.

ArrayRemove

지정된 인덱스로 시작하는 배열에서 지정된 수의 요소를 제거

```
bool  ArrayRemove(
   void&         array[],            // 모든 유형의 배열
   uint          start,              // 인덱스 제거를 시작
   uint          count=WHOLE_ARRAY   // 요소의 수
   );
```

매개변수

array[]

[in][out] 배열.

시작

[in] 배열 요소가 제거되는 시작 인덱스.

count=WHOLE_ARRAY

[in] 제거된 요소의 수. WHOLE_ARRAY 값은 배열 끝의 지정된 인덱스에서 모든 요소를 제거하는 것을 의미합니다.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류에 대한 정보를 가져오려면 GetLastError() 기능을 호출. 가능한 오류:

- 5052 - ERR_SMALL_ARRAY (너무 큰 *시작* 값),
- 5056 - ERR_SERIES_ARRAY (배열을 변경할 수 없습니다, 지표 버퍼),
- 4003 - ERR_INVALID_PARAMETER (너무 큰 *카운트* 값),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (소멸기가 있는 복잡한 개체를 포함하는 고정 크기 배열),
- 4006 - ERR_INVALID_ARRAY (소멸기가 있는 구조 또는 클래스 개체를 포함하는 고정 크기 배열).

참고

함수가 고정 크기 배열에 사용되는 경우 배열 크기가 변경되지 않고 나머지 "끝부분"이 *시작* 포지션으로 물리적으로 복사됩니다. 기능의 작동 방식을 정확하게 이해하려면 아래 예를 참조하십시오. "물리적" 복사는 복사된 개체가 생성자 또는 복사 연산자에 의해 생성되지 않음을 의미합니다. 대신 개체의 바이너리 표시가 복사됩니다. 따라서 ArrayRemove() 기능을 소멸기가 있는 개체가 포함된 고정 크기 배열에 적용할 수 없습니다 (ERR_INVALID_ARRAY 또는 ERR_STRUCT_WITHOBJECTS_ORCLASS 오류가 활성화됨). 이러한 개체를 제거할 때는 원래 개체와 해당 복사본에 대해 소멸기를 두 번 호출해야 합니다.

SetIndexBuffer() 기능으로 지표 버퍼로 지정된 동적 배열에서는 요소를 제거할 수 없습니다. 이로 인해 ERR_SERIES_ARRAY 오류가 발생합니다. 지표 버퍼의 경우 터미널의 실행 하위 시스템에 의해 모든 크키 변경 작업이 수행됩니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- 고정 크기 배열을 선언하고 값을 채우기
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- 요소를 제거하기 전에 배열을 표시
   Print("ArrayRemove() 호출 전");
   ArrayPrint(array);
//--- 배열에서 요소 2개를 삭제하고 새 집합을 표시
   ArrayRemove(array,4,2);
   Print("ArrayRemove() 호출 후");
   ArrayPrint(array);
/*
  실행 결과:
  ArrayRemove() 호출 전
   0 1 2 3 4 5 6 7 8 9
  ArrayRemove() 호출 후
   0 1 2 3 6 7 8 9 8 9
*/
```

추가 참조

ArrayInsert, ArrayCopy, ArrayResize, ArrayFree

© 2000-2025, MetaQuotes Ltd.

ArrayReverse

지정된 인덱스로 시작하는 배열의 지정된 요소 수를 반대로 적용.

```
bool  ArrayReverse(
   void&         array[],            // 모든 유형의 배열
   uint          start=0,            // 배열 역방향 시작하는 인덱스
   uint          count=WHOLE_ARRAY   // 요소의 수
   );
```

매개변수

array[]

[in][out] 배열.

start=0

[in] 인덱스 배열의 반전은 다음에서 시작.

count=WHOLE_ARRAY

[in] 반전된 요소의 수. WHOLE_ARRAY이면, 모든 배열 요소가 지정된 *start* 인덱스에서 시작하여 배열 끝까지 역방향으로 이동합니다.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환.

참고

ArraySetAsSeries() 기능은 배열 요소를 물리적으로 이동하지 않습니다. 대신, 인덱스 방향만 거꾸로 변경하여 시계열에서처럼 요소에 대한 액세스를 정렬합니다. ArrayReverse() 함수는 배열 요소를 물리적으로 이동하여 배열이 "역방향"이 되도록 합니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 고정 크기 배열을 선언하고 값을 채우기
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- 요소를 되돌리기 전에 배열을 표시
   Print("ArrayReverse() 호출 전");
   ArrayPrint(array);
//--- 배열 요소 3개를 역순으로 배치하고 새 집합을 표시
   ArrayReverse(array,4,3);
   Print("ArrayReverse() 호출 후");
   ArrayPrint(array);
```

© 2000-2025, MetaQuotes Ltd.

```
/*
   실행 결과:
   ArrayReverse() 호출 전
     0 1 2 3 4 5 6 7 8 9
   ArrayReverse() 호출 후
     0 1 2 3 6 5 4 7 8 9
*/
```

추가 참조

ArrayInsert, ArrayRemove, ArrayCopy, ArrayResize, ArrayFree, ArrayGetAsSeries, ArraySetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArraySetAsSeries

이 함수는 AS_SERIES 플래그를 선택된 동적 배열의 개체로 설정하고 요소는 시계열처럼 인덱싱.

```
bool  ArraySetAsSeries(
   const void&  array[],     // 참조에 의한 배열
   bool         flag         // true는 인덱스의 역순을 나타냅니다
   );
```

매개변수

array[]

[in][out] 설정할 숫자 배열.

플래그

[in] 배열 인덱싱 방향.

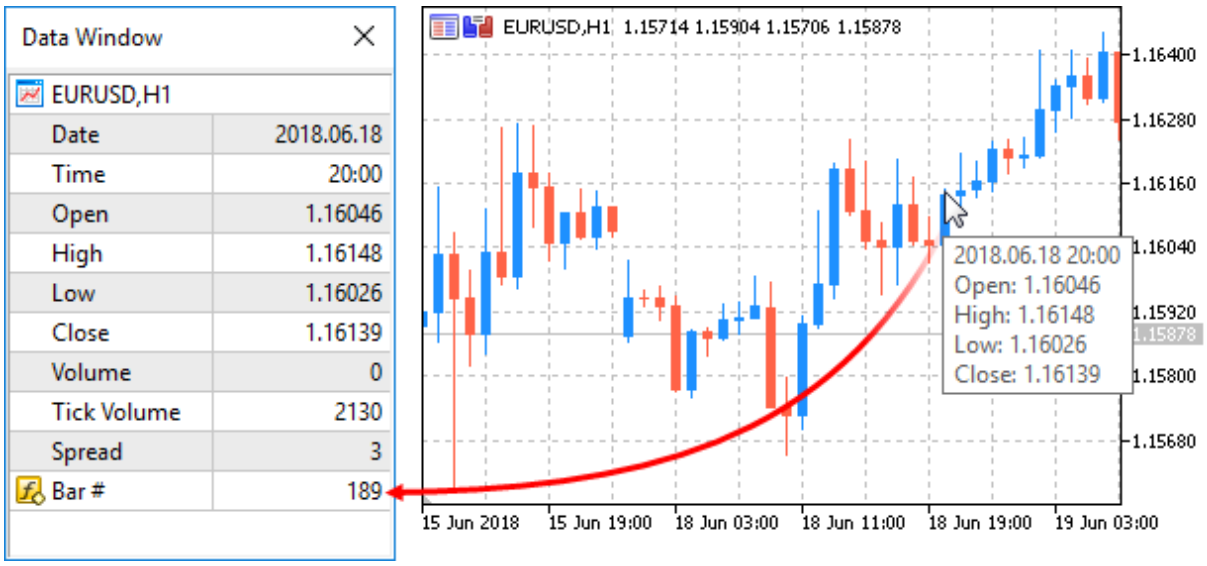
반환 값

이 함수는 성공 시 true를, 그렇지 않으면 false를 반환합니다.

참고

다차원 배열 또는 정적 배열(대괄호 안의 크기가 이미 컴파일 단계에서 미리 설정되어 있는 배열)에는 AS_SERIES 플래그를 설정할 수 없습니다. 시계열 인덱싱은 시계열 요소가 마지막부터 시작까지(최신 데이터부터 가장 오래된 데이터까지) 인덱싱된다는 점에서 공통 배열과 다릅니다.

예: 막대 번호를 표시하는 지표




```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- 플롯 수
#property indicator_label1  "숫자"
#property indicator_type1   DRAW_LINE
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_color1  CLR_NONE
//--- 지표 버퍼
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- 시계열에서처럼 버퍼에 대한 인덱싱 설정
   ArraySetAsSeries(NumerationBuffer,true);
//--- DataWindow에 표시되는 정확도 설정
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- 지표 배열 이름이 DataWindow에 표시되는 방법
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---  현재 0 막대의 개장 시간을 저장하겠습니다
   static datetime currentBarTimeOpen=0;
//--- time[] 배열로 액세스 되돌리기 - 시계열처럼 실행
   ArraySetAsSeries(time,true);
//--- 0 막대의 시간이 저장된 시간과 다를 경우
   if(currentBarTimeOpen!=time[0])
     {
     //--- 현재에서 차트 깊이까지 모든 막대를 열거
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

추가 참조

© 2000-2025, MetaQuotes Ltd.

시계열에 액세스, ArrayGetAsSeries

© 2000-2025, MetaQuotes Ltd.

ArraySize

함수는 선택된 배열의 요소 수를 반환.

```
int  ArraySize(
   const void&  array[]    // 확인된 배열
   );
```

매개변수

array[]

[in] 모든 유형의 배열.

반환 값

int 유형의 값.

참고

1차원 배열의 경우 ArraySize 에서 반환할 값은 ArrayRange(array,0)과 같습니다.

예:

```
void OnStart()
  {
//--- 배열 생성
   double one_dim[];
   double four_dim[][10][5][2];
//--- 크기
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- 보조 변수
   int size;
//--- 백업 없이 할당된 메모리
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. 1차원 배열
   Print("+================================================================+");
   Print("배열 크기:");
   Print("1. 1차원 배열");
   size=ArraySize(one_dim);
   PrintFormat("0차원 크기 = %d, Array size = %d",one_dim_size,size);
//--- 2. 다차원 배열
   Print("2 다차원 배열");
   size=ArraySize(four_dim);
   PrintFormat("0차원 크기 = %d, Array size = %d",four_dim_size,size);
//--- 차원 크기
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
   Print("확인:");
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("0차원 = 배열 크기 / (첫 번째 차원 * 두 번째 차원 * 세 번째 차원)");
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. 메모리 백업이 포함된 1차원 배열
   Print("3. 메모리 백업이 포함된 1차원 배열");
//--- 두 배 값
   one_dim_size*=2;
//--- 백업이 포함된 메모리 할당
   ArrayResize(one_dim,one_dim_size,reserve);
//--- 크기 인쇄
   size=ArraySize(one_dim);
   PrintFormat("백업이 있는 크기 = %d, 실제 배열 크기 = %d",one_dim_size+reserve,size);
  }
```

© 2000-2025, MetaQuotes Ltd.

ArraySort

다차원 숫자 배열의 첫 번째 차원에 있는 값을 오름차순으로 정렬..

```
bool  ArraySort(
   void&  array[]      // 정렬할 배열
   );
```

매개변수

array[]

[in][out] 정렬할 숫자 배열.

반환 값

이 함수는 성공 시 true를, 그렇지 않으면 false를 반환합니다.

참고

배열은 AS_SERIES 플래그 값에 관계없이 항상 오름차순으로 정렬됩니다.

ArraySort 기능 및 ArrayBSearch 에서는 모든 차원 배열을 매개변수로 사용할 수 있습니다. 그러나 검색 및 정렬은 상상 첫 번째(0) 차원에 적용됩니다.

예:

```
#property description "지표는 지난 달의 데이터를 분석하고 크기가 작은 모든 캔들스틱을 그립니다"
#property description "그리고 큰 틱 볼륨. 틱 볼륨 배열이 정렬됩니다"
#property description "그러한 캔들스틱을 정의하기 위해. 첫 번째 InpSmallVolume으로 구성된 볼
#property description "배열의 백분율이 작은 것으로 간주됩니다. 틱 볼륨이 있는 캔들스틱이 구성됨
#property description "배열의 마지막 InpBigVolume 백분율이 큰 것으로 간주됩니다."
//--- 지표 설정
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- 플롯
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- 사전정의된 상수
#define INDICATOR_EMPTY_VALUE 0.0
//--- 매개변수 입력
input int InpSmallVolume=15; // 작은 볼륨의 백분율 값 (<50)
input int InpBigVolume=20;   // 큰 볼륨의 백분율 값 (<50)
//--- 분석 시작 시간(이동 예정)
datetime ExtStartTime;
//--- 지표 버퍼
double   ExtOpenBuff[];
double   ExtHighBuff[];
double   ExtLowBuff[];
```

© 2000-2025, MetaQuotes Ltd.

```
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- 캔들스틱 표시를 위한 볼륨 경계 값
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------+
//| 틱 볼륨에 대한 테두리 값 수신                                    |
//+------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- 변수
   datetime stop_time;  // 종료 시간 복사
   long     buff[];     // 복사할 버퍼
//--- 종료 시간이 현재 시간입니다
   stop_time=TimeCurrent();
//--- 시작 시간이 현재 시간보다 한 달 빠릅니다
   ExtStartTime=GetStartTime(stop_time);
//--- 틱 볼륨 값 수신
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- 데이터 수신 실패, 재연산 명령을 시작하기 위해 false를 반환
      PrintFormat("틱 볼륨 값을 수신하는데 실패. 오류 코드 = %d",GetLastError());
      return(false);
     }
//--- 배열 크기 계산
   int size=ArraySize(buff);
//--- 배열을 정리
   ArraySort(buff);
//--- 틱 볼륨 좌측 및 우측 테두리 값 정의
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- 실행 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 전달된 데이터보다 한 달 적은 데이터 수신                         |
//+------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- 종료시간을 MqlDateTime 유형 구조 변수로 변환
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- 한 달 적은 데이터를 수신
   if(temp.mon>1)
      temp.mon-=1;  // 현재 달이 연도의 첫 번째 달이 아니므로 이전 달의 수가 하나 줄어듭니다
   else
     {
      temp.mon=12;  // 현재 달은 연도의 첫 번째 달이므로 이전 달의 수는 12,
```

© 2000-2025, MetaQuotes Ltd.

```
      temp.year-=1; // 연수가 하나 줄면
     }
//--- 일수는 28을 초과하지 않습니다
   if(temp.day>28)
      temp.day=28;
//--- 기일을 반환
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 입력 매개변수가 조건을 충족하는지 확인
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
     {
      Print("잘못된 입력 매개변수");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- 표시하지 않을 값을 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- 지표 버퍼에 대한 라벨 설정
   PlotIndexSetString(0,PLOT_LABEL,"시가; 고가; 저가; 종가");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 아직 처리되지 않은 바가 있는지 확인
   if(prev_calculated<rates_total)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 볼륨에 대한 우측 및 좌측 테두리의 새 값을 수신
      if(!GetVolumeBorders())
         return(0);
     }
//--- 막대 계산을 위한 시작 변수
   int start=prev_calculated;
//--- 지표 값이 이전 틱에서 이미 계산된 경우 마지막 막대에서 작업
   if(start>0)
      start--;
//--- 시계열에서 직접 인덱싱 설정
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- 지표 값의 계산 루프
   for(int i=start;i<rates_total;i++)
     {
      //--- 초기 데이터 시작부터 캔들스틱을 채움
      if(ExtStartTime<=time[i])
        {
         //--- 값이 우측 테두리보다 작지 않으면 캔들스틱을 채움
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- 캔들스틱을 그리기 위한 데이터 수신
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- DodgerBlue 색상
            ExtColorBuff[i]=0;
            //--- 루프를 계속
            continue;
           }
         //--- 값이 좌측 테두리를 초과하지 않으면 캔들스틱을 채움
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //--- 캔들스틱을 그리기 위한 데이터 수신
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- 주황색 색상
            ExtColorBuff[i]=1;
            //--- 루프를 계속
            continue;
           }
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 계산에 포함되지 않은 막대의 빈 값을 설정
      ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

추가 참조

ArrayBsearch

© 2000-2025, MetaQuotes Ltd.

ArraySwap

동일한 유형의 두 동적 배열의 내용을 스왑. 다차원 배열의 경우 첫 번째 요소를 제외한 모든 차원의 요소 수가 일치해야 합니다.

```
bool  ArraySwap(
   void&  array1[],      // 첫 번째 배열
   void&  array2[]       // 두 번째 배열
   );
```

매개변수

array1[]

[in][out] 숫자 유형의 배열.

array2[]

[in][out] 숫자 유형의 배열.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 이 경우, GetLastError()는 ERR_INVALID_ARRAY 에러 코드를 반환합니다.

참고

함수는 첫 번째 배열을 제외하고 유형이 동일하고 크기가 동일한 동적 배열을 허용합니다. 정수 유형의 경우 기호는 무시됩니다. 즉, char==uchar)

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 시세를 저장할 배열
   double source_array[][8];
   double   dest_array[][8];
   MqlRates rates[];
//--- 현재 시간 프레임에 대한 최근 20개의 초 데이터를 얻습니다.
   int copied=CopyRates(NULL,0,0,20,rates);
   if(copied<=0)
     {
      PrintFormat("CopyRates(%s,0,0,20,rates) failed, error=%d",
                  Symbol(),GetLastError());
      return;
     }
//--- 복사된 데이터의 양에 대한 배열 크기 설정
   ArrayResize(source_array,copied);
//--- rate_array_1[] 배열을 rates[]의 데이터로 채움
   for(int i=0;i<copied;i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      source_array[i][0]=(double)rates[i].time;
      source_array[i][1]=rates[i].open;
      source_array[i][2]=rates[i].high;
      source_array[i][3]=rates[i].low;
      source_array[i][4]=rates[i].close;
      source_array[i][5]=(double)rates[i].tick_volume;
      source_array[i][6]=(double)rates[i].spread;
      source_array[i][7]=(double)rates[i].real_volume;
     }
//--- source_array[] 및 dest_array[] 간에 데이터 스왑
   if(!ArraySwap(source_array,dest_array))
     {
      PrintFormat("ArraySwap(source_array,rate_array_2) failed, error code=%d",GetLast
      return;
     }
//--- 스왑 후 원본 배열이 0이 되었는지 확인
   PrintFormat("ArraySwap() done: ArraySize(source_array)=%d",ArraySize(source_array))
//--- dest_array[] 대상 배열의 데이터 표시
   ArrayPrint(dest_array);
  }
```

추가 참조

ArrayCopy, ArrayFill, ArrayRange, ArrayIsDynamic

© 2000-2025, MetaQuotes Ltd.

ArrayToFP16

float 또는 double 유형의 배열을 주어진 형식으로 ushort 유형의 배열로 복사합니다.

```
bool   ArrayToFP16(
   const ushort&         dst_array[],        // copy to
   const float&          src_array[],        // copy from
   ENUM_FLOAT16_FORMAT   fmt                 // format
   );
```

double 유형에 대한 오버로딩

```
bool   ArrayToFP16(
   const ushort&         dst_array[],        // copy to
   const double&         src_array[],        // copy from
   ENUM_FLOAT16_FORMAT   fmt                 // format
   );
```

매개 변수

dst_array[]

[out] 수신 배열 또는 ushort 타입.

src_array[]

[in] float 또는 double 유형의 소스 배열.

fmt

[in] ENUM_FLOAT16_FORMAT 열거에서 형식을 복사.

반환값

성공하면 true를, 그렇지 않으면 false를 반환.

참조

FLOAT16 및 BFLOAT16 형식은 ENUM_FLOAT16_FORMAT 열거에 정의되어 있으며 MQL5에서 ONNX 모델을 사용한 작업에만 사용됩니다.

이 함수는 float 또는 double 유형의 입력 매개변수를 FLOAT16 및 BFLOAT16 유형으로 변환합니다. 이러한 입력 매개변수는 OnnxRun 함수에서 사용됩니다.

FLOAT16 은 반정밀도 float라고도 하며 16비트를 사용하여 부동 소수점 숫자를 나타냅니다. 이 형식은 정확성과 계산 효율성 간의 균형을 제공합니다. FLOAT16은 대규모 데이터 세트의 고성능 처리가 필요한 딥러닝 알고리즘과 신경망에 널리 사용됩니다. 이 형식은 숫자의 크기를 줄여 계산을 가속화합니다. 이는 GPU에서 심층 신경망을 훈련할 때 특히 중요합니다.

BFLOAT16(또는 Brain Floating Point 16)도 16비트를 사용하지만 형식 표현 방식에서 FLOAT16과는 다릅니다. 이 형식에서는 8비트가 지수를 표현하기 위해 할당되고 나머지 7비트는 가수를 표현하기 위해 사용됩니다. 이 형식은 딥 러닝 및 인공 지능, 특히 Google의 TPU(Tensor Processor Unit)에 사용하기 위해 개발되었습니다. BFLOAT16은 신경망 훈련에서 뛰어난 성능을 보여주며 계산 속도를 효과적으로 가속화할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

예: 다음 기사의 함수 float16 및 float8 형식으로 ONNX 모델 작업

```
//+------------------------------------------------------------------+
//| RunCastFloat16ToDouble                                           |
//+------------------------------------------------------------------+
bool RunCastFloat16ToDouble(long model_handle)
  {
   PrintFormat("test=%s",__FUNCTION__);
   double test_data[12]= {1,2,3,4,5,6,7,8,9,10,11,12};
   ushort data_uint16[12];
   if(!ArrayToFP16(data_uint16,test_data,FLOAT_FP16))
     {
      Print("error in ArrayToFP16. error code=",GetLastError());
      return(false);
     }
   Print("test array:");
   ArrayPrint(test_data);
   Print("ArrayToFP16:");
   ArrayPrint(data_uint16);
   U<ushort> input_float16_values[3*4];
   U<double> output_double_values[3*4];
   float test_data_float[];
   if(!ArrayFromFP16(test_data_float,data_uint16,FLOAT_FP16))
     {
      Print("error in ArrayFromFP16. error code=",GetLastError());
      return(false);
     }
   for(int i=0; i<12; i++)
     {
      input_float16_values[i].value=data_uint16[i];
      PrintFormat("%d input value =%f  Hex float16 = %s  ushort value=%d",i,test_data_
     }
   Print("ONNX input array:");
   ArrayPrint(input_float16_values);
   bool res=OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float16_values,output_double
   if(!res)
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      return(false);
     }
   Print("ONNX output array:");
   ArrayPrint(output_double_values);
//---
   double sum_error=0.0;
   for(int i=0; i<12; i++)
     {
      double delta=test_data[i]-output_double_values[i].value;
```

© 2000-2025, MetaQuotes Ltd.

```
      sum_error+=MathAbs(delta);
      PrintFormat("%d output double %f = %s  difference=%f",i,output_double_values[i],
     }
//---
   PrintFormat("test=%s   sum_error=%f",__FUNCTION__,sum_error);
//---
   return(true);
  }
```

참조

ArrayFromFP16, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayToFP8

float 또는 double 유형의 배열을 지정된 형식으로 uchar 유형의 배열로 복사합니다.

```
bool   ArrayToFP8(
   const uchar&          dst_array[],        // copy to
   const float&          src_array[],        // copy from
   ENUM_FLOAT8_FORMAT    fmt                 // format
   );
```

double 유형에 대한 오버로딩

```
bool   ArrayToFP8(
   const uchar&          dst_array[],        // copy to
   const double&         src_array[],        // copy from
   ENUM_FLOAT8_FORMAT    fmt                 // format
   );
```

매개 변수

dst_array[]

[out] uchar 유형의 수신 배열.

src_array[]

[in] float 또는 double 유형의 소스 배열.

fmt

[in] ENUM_FLOAT8_FORMAT 열거에서 형식을 복사.

반환값

성공하면 true를, 그렇지 않으면 false를 반환.

참조

모든 종류의 FP8 형식은 ENUM_FLOAT8_FORMAT 열거에 정의되어 있으며 MQL5에서 ONNX 모델과의 작업에만 사용됩니다.

이 함수는 float 또는 double 유형의 입력 매개변수를 FP8 유형 중 하나로 변환합니다. 이러한 입력 매개변수는 OnnxRun 함수에서 사용됩니다.

FP8(8비트 부동 소수점)은 부동 소수점 자리를 나타내는 데 사용되는 데이터 유형 중 하나입니다. FP8에서 각 숫자는 8개의 데이터 비트로 표시되며 일반적으로 부호, 지수 및 가수의 세 가지 구성 요소로 나뉩니다. 이 형식은 정확성과 저장 효율성 사이의 균형을 제공하므로 메모리 용량과 계산 효율성이 필요한 애플리케이션에 적합합니다.

FP8은 간결한 숫자 표현을 사용하여 메모리의 양을 줄이고 계산을 가속화합니다. 또한 FP8은 산술 계산 및 신호 처리와 같은 로 레벨의 작업을 구현하는 데 유용할 수 있습니다.

예: 다음 기사의 함수 float16 및 float8 형식으로 ONNX 모델 작업

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| RunCastFloat8Float                                               |
//+------------------------------------------------------------------+
bool RunCastFloat8ToFloat(long model_handle,const ENUM_FLOAT8_FORMAT fmt)
  {
   PrintFormat("TEST: %s(%s)",__FUNCTION__,EnumToString(fmt));
//---
   float test_data[15]   = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
   uchar data_float8[15] = {};
   if(!ArrayToFP8(data_float8,test_data,fmt))
     {
      Print("error in ArrayToFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   U<uchar> input_float8_values[3*5];
   U<float> output_float_values[3*5];
   float    test_data_float[];
//--- convert float8 to float
   if(!ArrayFromFP8(test_data_float,data_float8,fmt))
     {
      Print("error in ArrayFromFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   for(uint i=0; i<data_float8.Size(); i++)
     {
      input_float8_values[i].value=data_float8[i];
      PrintFormat("%d input value =%f  Hex float8 = %s  ushort value=%d",i,test_data_f
     }
   Print("ONNX input array: ",ArrayToString(input_float8_values));
//--- execute model (convert float8 to float using ONNX)
   if(!OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float8_values,output_float_values
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   Print("ONNX output array: ",ArrayToString(output_float_values));
//--- calculate error (compare ONNX and ArrayFromFP8 results)
   double sum_error=0.0;
   for(uint i=0; i<test_data.Size(); i++)
     {
      double delta=test_data_float[i]-(double)output_float_values[i].value;
      sum_error+=MathAbs(delta);
      PrintFormat("%d output float %f = %s difference=%f",i,output_float_values[i].val
     }
//---
   PrintFormat("%s(%s): sum_error=%f\n",__FUNCTION__,EnumToString(fmt),sum_error);
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

참조

ArrayFromFP8, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayFromFP16

Copies an array of type ushort into an array of float or double type with the given format.

```
bool   ArrayFromFP16(
   const float&          dst_array[],        // copy to
   const ushort&         src_array[],        // copy from
   ENUM_FLOAT16_FORMAT   fmt                 // format
   );
```

double 유형에 대한 오버로딩

```
bool   ArrayFromFP16(
   const double&         dst_array[],        // copy to
   const ushort&         src_array[],        // copy from
   ENUM_FLOAT16_FORMAT   fmt                 // format
   );
```

매개 변수

dst_array[]

[out] Receiver array or type float or double.

src_array[]

[in] Source array of type ushort.

fmt

[in] ENUM_FLOAT16_FORMAT 열거에서 형식을 복사.

반환값

성공하면 true를, 그렇지 않으면 false를 반환.

참조

FLOAT16 및 BFLOAT16 형식은 ENUM_FLOAT16_FORMAT 열거에 정의되어 있으며 MQL5에서 ONNX 모델을 사용한 작업에만 사용됩니다.

OnnxRun 함수 실행에서 얻은 출력 매개변수가 FLOAT16 및 BFLOAT16 유형인 경우 이 함수를 사용하여 결과를 부동 또는 더블 배열로 변환할 수 있습니다.</t2>

FLOAT16 은 반정밀도 float라고도 하며 16비트를 사용하여 부동 소수점 숫자를 나타냅니다. 이 형식은 정확성과 계산 효율성 간의 균형을 제공합니다. FLOAT16은 대규모 데이터 세트의 고성능 처리가 필요한 딥러닝 알고리즘과 신경망에 널리 사용됩니다. 이 형식은 숫자의 크기를 줄여 계산을 가속화합니다. 이는 GPU에서 심층 신경망을 훈련할 때 특히 중요합니다.

BFLOAT16(또는 Brain Floating Point 16)도 16비트를 사용하지만 형식 표현 방식에서 FLOAT16과는 다릅니다. 이 형식에서는 8비트가 지수를 표현하기 위해 할당되고 나머지 7비트는 가수를 표현하기 위해 사용됩니다. 이 형식은 딥 러닝 및 인공 지능, 특히 Google의 TPU(Tensor Processor Unit)에 사용하기 위해 개발되었습니다. BFLOAT16은 신경망 훈련에서 뛰어난 성능을 보여주며 계산 속도를 효과적으로 가속화할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

예: 기사의 함수 float16 및 float8 형식으로 ONNX 모델 작업

```
//+------------------------------------------------------------------+
//| RunCastFloat16ToDouble                                           |
//+------------------------------------------------------------------+
bool RunCastFloat16ToDouble(long model_handle)
  {
   PrintFormat("test=%s",__FUNCTION__);
   double test_data[12]= {1,2,3,4,5,6,7,8,9,10,11,12};
   ushort data_uint16[12];
   if(!ArrayToFP16(data_uint16,test_data,FLOAT_FP16))
     {
      Print("error in ArrayToFP16. error code=",GetLastError());
      return(false);
     }
   Print("test array:");
   ArrayPrint(test_data);
   Print("ArrayToFP16:");
   ArrayPrint(data_uint16);
   U<ushort> input_float16_values[3*4];
   U<double> output_double_values[3*4];
   float test_data_float[];
   if(!ArrayFromFP16(test_data_float,data_uint16,FLOAT_FP16))
     {
      Print("error in ArrayFromFP16. error code=",GetLastError());
      return(false);
     }
   for(int i=0; i<12; i++)
     {
      input_float16_values[i].value=data_uint16[i];
      PrintFormat("%d input value =%f  Hex float16 = %s  ushort value=%d",i,test_data_
     }
   Print("ONNX input array:");
   ArrayPrint(input_float16_values);
   bool res=OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float16_values,output_double
   if(!res)
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      return(false);
     }
   Print("ONNX output array:");
   ArrayPrint(output_double_values);
//---
   double sum_error=0.0;
   for(int i=0; i<12; i++)
     {
      double delta=test_data[i]-output_double_values[i].value;
```

© 2000-2025, MetaQuotes Ltd.

```
      sum_error+=MathAbs(delta);
      PrintFormat("%d output double %f = %s  difference=%f",i,output_double_values[i],
     }
//---
   PrintFormat("test=%s   sum_error=%f",__FUNCTION__,sum_error);
//---
   return(true);
  }
```

참조

ArrayToFP16, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

ArrayFromFP8

uchar 유형의 배열을 지정된 형식의 float 또는 double 유형의 배열로 복사합니다.

```
bool   ArrayFromFP8(
   const float&          dst_array[],        // copy to
   const uchar&          src_array[],        // copy from
   ENUM_FLOAT8_FORMAT    fmt                 // format
   );
```

double 유형에 대한 오버로딩

```
bool   ArrayFromFP8(
   const double&         dst_array[],        // copy to
   const uchar&          src_array[],        // copy from
   ENUM_FLOAT8_FORMAT    fmt                 // format
   );
```

매개 변수

dst_array[]

[out] 수신기 배열 또는 유형은 float 또는 double.

src_array[]

[in] uchar 유형의 소스 배열.

fmt

[in] ENUM_FLOAT8_FORMAT 열거에서 형식을 복사.

반환값

성공하면 true를, 그렇지 않으면 false를 반환.

참조

모든 종류의 FP8 형식은 ENUM_FLOAT8_FORMAT 열거에 정의되어 있으며 MQL5에서 ONNX 모델과의 작업에만 사용됩니다.

OnnxRun 함수 실행에서 얻은 출력 매개변수가 ENUM_FLOAT8_FORMAT 열거의 FP8인 경우 이 함수를 사용하여 결과를 부동 또는 더블 배열로 변환할 수 있습니다.

FP8(8비트 부동 소수점)은 부동 소수점 자리를 나타내는 데 사용되는 데이터 유형 중 하나입니다. FP8에서 각 숫자는 8개의 데이터 비트로 표시되며 일반적으로 부호, 지수 및 가수의 세 가지 구성 요소로 나뉩니다. 이 형식은 정확성과 저장 효율성 사이의 균형을 제공하므로 메모리 용량과 계산 효율성이 필요한 애플리케이션에 적합합니다.

FP8은 간결한 숫자 표현을 사용하여 메모리의 양을 줄이고 계산을 가속화합니다. 또한 FP8은 산술 계산 및 신호 처리와 같은 로 레벨의 작업을 구현하는 데 유용할 수 있습니다.

예: 다음 기사의 함수 float16 및 float8 형식으로 ONNX 모델 작업

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| RunCastFloat8Float                                               |
//+------------------------------------------------------------------+
bool RunCastFloat8ToFloat(long model_handle,const ENUM_FLOAT8_FORMAT fmt)
  {
   PrintFormat("TEST: %s(%s)",__FUNCTION__,EnumToString(fmt));
//---
   float test_data[15]   = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
   uchar data_float8[15] = {};
   if(!ArrayToFP8(data_float8,test_data,fmt))
     {
      Print("error in ArrayToFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   U<uchar> input_float8_values[3*5];
   U<float> output_float_values[3*5];
   float    test_data_float[];
//--- convert float8 to float
   if(!ArrayFromFP8(test_data_float,data_float8,fmt))
     {
      Print("error in ArrayFromFP8. error code=",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   for(uint i=0; i<data_float8.Size(); i++)
     {
      input_float8_values[i].value=data_float8[i];
      PrintFormat("%d input value =%f  Hex float8 = %s  ushort value=%d",i,test_data_f
     }
   Print("ONNX input array: ",ArrayToString(input_float8_values));
//--- execute model (convert float8 to float using ONNX)
   if(!OnnxRun(model_handle,ONNX_NO_CONVERSION,input_float8_values,output_float_values
     {
      PrintFormat("error in OnnxRun. error code=%d",GetLastError());
      OnnxRelease(model_handle);
      return(false);
     }
   Print("ONNX output array: ",ArrayToString(output_float_values));
//--- calculate error (compare ONNX and ArrayFromFP8 results)
   double sum_error=0.0;
   for(uint i=0; i<test_data.Size(); i++)
     {
      double delta=test_data_float[i]-(double)output_float_values[i].value;
      sum_error+=MathAbs(delta);
      PrintFormat("%d output float %f = %s difference=%f",i,output_float_values[i].val
     }
//---
   PrintFormat("%s(%s): sum_error=%f\n",__FUNCTION__,EnumToString(fmt),sum_error);
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
```

참조

ArrayToFP8, ArrayCopy

© 2000-2025, MetaQuotes Ltd.

행렬과 벡터

행렬은 double, float 또는 복소수의 2차원 배열입니다.

벡터는 double, float 또는 복소수의 1차원 배열입니다. 벡터는 벡터가 수직인지 수평인지 구분하지 않습니다. 수직인지 수평인지의 구분은 사용 컨텍스트에서 결정됩니다. 예를 들어 벡터 연산 Dot의 경우 왼쪽 벡터가 수평이고 오른쪽 벡터가 수직이라고 가정합니다. 유형 표시가 필요한 경우 1행 또는 1열 행렬을 사용할 수 있습니다. 하지만 이것은 일반적으로 필요하지 않습니다.

행렬과 벡터는 데이터를 저장하는 메모리를 동적으로 할당합니다. 그러나 행렬과 벡터는 차원이 있는 데이터 유형과 같은 특정한 속성을 가진 객체입니다. 행렬 및 벡터 속성은 vector_a.Size(), matrix_b.Rows(), vector_c.Norm(), matrix_d.Cond() 등과 같은 메서드를 사용하여 얻을 수 있습니다. 모든 차원은 변경될 수 있습니다.

행렬을 생성하고 초기화할 때는 소위 정적 메소드가 사용됩니다(이는 클래스의 정적 메소드와 유사함). 예: matrix::Eye(), matrix::Identity(), matrix::Ones(), vector::Ones(), matrix::Zeros(), vector::Zeros(), matrix::Full(), vector::Full(), matrix::Tri().

현재로서는 이 개발 방향이 아직 완성되지 않습니다. 이로 인해 행렬 및 벡터 연산은 복합 데이터 유형의 사용을 의미하지 않습니다.

MQL5는 행렬 및 벡터를 DLL로 전달하는 것을 지원합니다. 이를 통해 외부 변수에서 관련 유형을 활용하는 함수를 가져올 수 있습니다.

행렬과 벡터는 버퍼에 대한 포인터로 DLL에 전달됩니다. 예를 들어 float 유형의 행렬을 전달하려면 DLL에서 내보낸 함수의 해당 매개변수가 float 유형의 버퍼 포인터를 가져야 합니다.

MQL5

```
#import "mmlib.dll"
bool sgemm(uint flags, matrix<float> &C, const matrix<float> &A, const matrix<float> &
#import
```

C++

```
extern "C" __declspec(dllexport) bool sgemm(UINT flags, float *C, const float *A, cons
```

버퍼 외에도 올바른 처리를 위해 행렬 및 벡터의 크기를 전달해야 합니다.

모든 행렬 및 벡터의 메서드는 아래에 알파벳순으로 나열되어 있습니다.

함수	액션	카테고리
Activation	활성화 함수 값을 계산하고 전달된 벡터/행렬에 씁니다.	머신 러닝
ArgMax	최대값의 인덱스를 반환	통계
ArgMin	최소값의 인덱스를 반환	통계
ArgSort	정렬된 인덱스를 반환	조작
Assign	자동 캐스트로 행렬, 벡터 또는 배열을 복사	초기화

© 2000-2025, MetaQuotes Ltd.

함수	액션	카테고리
Average	행렬/벡터 값의 가중 평균 계산	통계
Cholesky	촐레스키 분해 계산	변환
Clip	행렬/벡터의 요소를 주어진 유효한 값 범위로 제한	조작
Col	열 벡터를 반환. 지정된 열에 벡터를 작성.	조작
Cols	행렬의 열 수 반환	기능
Compare	지정된 정밀함으로 두 행렬/벡터의 요소를 비교	조작
CompareByDigits	두 행렬/벡터의 요소를 유효 숫자 정밀도로 비교	조작
Cond	행렬의 조건수 계산	기능
Convolve	두 벡터의 불연속 선형 컨벌루션을 반환	곱
Copy	주어진 행렬/벡터의 복사본을 반환	조작
Concat	1 행렬에 2 개의 하위 매트릭스를 연결합니다. 하나의 벡터에 2개의 벡터를 연결합니다.	조작
CopyIndicatorBuffer	지정된 지표의 지정된 양의 버퍼를 벡터로 가져오기	초기화
CopyRates	지정된 양의 지정된 기호 기간의 MqlRates 구조의 과거 시리즈를 행렬 또는 벡터로 가져옵니다.	초기화
CopyTicks	틱을 MqlTick 구조에서 행렬 또는 벡터로 가져오기	초기화
CopyTicksRange	지정된 날짜 범위 내에서 MqlTick 구조에서 행렬 또는 벡터로 틱 가져오기	초기화
CorrCoef	Pearson 상관 계수 계산(선형 상관 계수)	곱
Correlate	두 벡터의 상호 상관을 계산	곱
Cov	공분산 행렬 계산	곱
CumProd	주어진 축을 따라 있는 요소를 포함하여 행렬/벡터 요소의 누적 곱을 반환	통계

© 2000-2025, MetaQuotes Ltd.

함수	액션	카테고리
CumSum	주어진 축을 따라 있는 요소를 포함하여 행렬/벡터 요소의 누적 합계를 반환	통계
Derivative	활성화 함수 도함수 값을 계산하고 이들을 전달된 벡터/행렬에 씁니다.	머신 러닝
Det	정사각형 가역 행렬의 행렬식을 계산	기능
Diag	대각선 추출 또는 대각선 행렬 구성	조작
Dot	두 벡터의 내적	곱
Eig	정사각형 행렬의 고유값과 오른쪽 고유벡터를 계산	변환
EigVals	일반 행렬의 고유값을 계산	변환
Eye	대각선에 1이 있고 이외의 곳에 0이 있는 행렬을 반환	초기화
Fill	지정된 값으로 기존 행렬 또는 벡터 채우기	초기화
Flat	두 개가 아닌 하나의 인덱스를 통해 행렬 요소에 액세스	조작
Full	주어진 값으로 채워진 새로운 행렬을 생성하고 반환	초기화
GeMM	GeMM(General Matrix Multiply) 메서드는 두 행렬의 일반 곱셈을 구현합니다.	곱
HasNan	행렬/벡터에서 NaN 수를 반환	조작
Hsplit	행렬을 여러 부분 행렬로 수평 분할 axis=0으로 Split과 동일	조작
Identity	지정된 크기의 항등 행렬을 생성	초기화
Init	행렬 또는 벡터 초기화	초기화
Inner	두 행렬의 내적	곱
Inv	Jordan-Gauss 메서드로 정사각형 가역 행렬의 곱셈 역행렬을 계산	솔루션
Kron	두 행렬, 행렬과 벡터, 벡터와 행렬 또는 두 벡터의 Kronecker 곱을 반환	곱

© 2000-2025, MetaQuotes Ltd.

함수	액션	카테고리
LinearRegression	계산된 선형 회귀 값으로 벡터/행렬을 계산합니다	통계
Loss	손실 함수 값을 계산하고 전달된 벡터/행렬에 씁니다.	머신 러닝
LstSq	선형 대수 방정식의 최소제곱해를 반환(비제곱 또는 축퇴 행렬의 경우).	솔루션
LU	행렬의 LU 분해 구현: 하부 삼각 행렬과 상부 삼각 행렬의 곱	변환
LUP	행 순열만 있는 LU 분해를 참조하는 부분 순열을 사용하여 LUP 분해를 구현: PA=LU	변환
MatMul	두 행렬의 행렬 곱	곱
Max	행렬/벡터의 최대값 반환	통계
Mean	요소 값의 산술 평균 계산	통계
Median	행렬/벡터 요소의 중앙값 계산	통계
Min	행렬/벡터의 최소값 반환	통계
Norm	행렬 또는 벡터 놈 반환	기능
Ones	1로 채워진 새 행렬을 만들고 반환	초기화
Outer	두 행렬 또는 두 벡터의 외적 계산	곱
Percentile	지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 지정된 백분위수를 반환	통계
PInv	Moore-Penrose 방법으로 행렬의 유사 역행렬을 계산	솔루션
Power	정사각 행렬을 정수 거듭제곱으로 올림	곱
Prod	주어진 축에 대해서도 실행할 수 있는 행렬/벡터 요소의 곱을 반환	통계
Ptp	행렬/벡터 또는 주어진 행렬 축의 값의 범위를 반환	통계
QR	행렬의 qr 인수분해를 계산	변환
Quantile	지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 지정된 분위수를 반환	통계

© 2000-2025, MetaQuotes Ltd.

함수	액션	카테고리
Random	정적 함수 랜덤 값으로 채워진 새로운 행렬 또는 벡터를 작성하고 반환하십시오. 램덤 값은 지정된 범위 내에서 균일하게 생성됩니다.	초기화
Rank	가우스 방법을 사용하여 행렬 순위 반환	기능
RegressionMetric	지정된 데이터 배열에 구성된 회귀선의 편차 오류로 회귀 메트릭을 계산	통계
Reshape	데이터를 변경하지 않고 행렬의 모양 변경	조작
Resize	모양과 크기가 변경된 새 행렬 반환	조작
Row	행 벡터를 반환	조작
Rows	행렬의 행 수를 반환	기능
Set	지정된 인덱스로 벡터 요소의 값을 정합니다	조작
Size	벡터의 크기를 반환	기능
SLogDet	행렬의 행렬식의 부호와 로그를 계산	기능
Solve	선형 행렬 방정식 또는 선형 대수 방정식 시스템 풀기	솔루션
Sort	place별로 정렬	조작
Spectrum	행렬곱 AT*A의 고유값 집합으로 행렬의 스펙트럼을 계산	기능
Split	행렬을 여러 부분 행렬로 분할	조작
Std	지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 표준 편차를 반환	통계
Sum	주어진 축(축)에 대해서 실행할 수 있는 행렬/벡터 요소의 합을 반환	통계
SVD	특이값 분해	변환
SwapCols	행렬의 열을 바꿈	조작
SwapRows	행렬에서 행을 서로 바꿈	조작
Trace	행렬의 대각선을 따라 합계를 반환	기능

© 2000-2025, MetaQuotes Ltd.

함수	액션	카테고리
Transpose	전치(축 교환) 및 수정된 행렬 반환	조작
Tri	지정된 대각선 이하에 1이 있고 다른 위치에 0이 있는 행렬을 구성	초기화
TriL	k번째 대각선 위의 요소가 0인 행렬의 복사본을 반환 Lower triangular matrix	조작
TriU	k 번째 대각선 아래의 요소가 0인 행렬의 복사본을 반환 Upper triangular matrix	조작
Var	행렬/벡터 요소 값의 분산 계산	통계
Vsplit	행렬을 세로로 여러 부분 행렬로 분할 axis=1로 Split과 동일	조작
Zeros	0으로 채워진 새 행렬을 만들고 반환	초기화

© 2000-2025, MetaQuotes Ltd.

행렬 및 벡터 유형

행렬 및 벡터는 선형 대수 연산을 가능하게 하는 MQL5의 특수한 데이터 유형입니다. 다음과 같은 데이터 유형이 있습니다:

- 행렬 — double 요소를 포함하는 행렬.
- matrixf — float 요소를 포함하는 행렬.
- matrixc — 복소수 요소를 포함하는 행렬.
- 벡터 — double 요소를 포함하는 벡터.
- vectorf — float 요소를 포함하는 벡터.
- vectorc — 복소수 요소를 포함하는 벡터.

템플릿 함수는 해당 유형 대신에 행렬<double>, 행렬<float>, 벡터<double>, 벡터<float>와 같은 표기법을 지원합니다.

행렬 및 벡터 초기화 메서드

함수	액션
Eye	대각선에 1이 있고 이외의 곳에 0이 있는 행렬을 반환
Identity	지정된 크기의 항등 행렬을 생성
Ones	1로 채워진 새 행렬을 만들고 반환
Zeros	0으로 채워진 새 행렬을 만들고 반환
Full	주어진 값으로 채워진 새로운 행렬을 생성하고 반환
Tri	주어진 대각선 이하에 1이 있고 다른 위치에 0이 있는 행렬을 구성
Init	행렬 또는 벡터를 초기화합니다.
Fill	지정된 값으로 기존 행렬 또는 벡터 채우기

© 2000-2025, MetaQuotes Ltd.

행렬 및 벡터 연산에 대한 열거형

이 섹션에서는 다양한 행렬 및 벡터 메서드에 사용되는 열거형에 대해 설명합니다.

ENUM_AVERAGE_MODE

평활화 타입 열거형

ID	설명
AVERAGE_NONE	평균화 없음 각 라벨에 대한 결과가 별도로 제공됨
AVERAGE_BINARY	이진 분류에 대한 라벨 1 결과
AVERAGE_MICRO	평균 오류 행렬 결과(혼동 행렬)
AVERAGE_MACRO	각 라벨의 오차 행렬 결과의 평균 결과
AVERAGE_WEIGHTED	가중 평균 결과

ENUM_VECTOR_NORM

vector::Norm에 대한 벡터 노름의 열거

ID	설명
VECTOR_NORM_INF	Inf norm
VECTOR_NORM_MINUS_INF	Minus Inf norm
VECTOR_NORM_P	Norm P

ENUM_MATRIX_NORM

matrix::Norm에 대한 행렬 노름의 열거와 matrix::Cond 행렬 조건을 얻기 위한 행렬 노름의 열거.

ID	설명
MATRIX_NORM_FROBENIUS	Frobenius norm
MATRIX_NORM_SPECTRAL	Spectral norm
MATRIX_NORM_NUCLEAR	Nuclear norm
MATRIX_NORM_INF	Inf norm
MATRIX_NORM_P1	P1 norm
MATRIX_NORM_P2	P2 norm

© 2000-2025, MetaQuotes Ltd.

ID	설명
MATRIX_NORM_MINUS_INF	Minus Inf norm
MATRIX_NORM_MINUS_P1	Minus P1 norm
MATRIX_NORM_MINUS_P2	Minus P2 norm

ENUM_VECTOR_CONVOLVE

변환 vector::Convolve을 위한 열거 와 cross-correlation vector::Correlate을 위한 열거.

ID	설명
VECTOR_CONVOLVE_FULL	Convolve full
VECTOR_CONVOLVE_SAME	Convolve same
VECTOR_CONVOLVE_VALID	Convolve valid

ENUM_REGRESSION_METRIC

vector::RegressionMetric을 위한 회귀 메트릭의 열거.

ID	설명
REGRESSION_MAE	평균 절대 오차
REGRESSION_MSE	평균 제곱 오차
REGRESSION_RMSE	평균 제곱근 오차
REGRESSION_R2	결정계수(R-Squared)
REGRESSION_MAPE	평균 절대(Mean Absolute) 백분율 오류
REGRESSION_MSPE	평균(Mean) 제곱 백분율 오류
REGRESSION_RMSLE	평균 제곱근 대수 오차
REGRESSION_SMAPE	대칭 평균(Mean) 절대 백분율 오류
REGRESSION_MAXE	최대 절대 오차
REGRESSION_MEDE	중앙값 절대 오차
REGRESSION_MPD	평균 포아송 편차
REGRESSION_MGD	평균 감마 편차
REGRESSION_EXPV	설명된 분산

© 2000-2025, MetaQuotes Ltd.

ENUM_CLASSIFICATION_METRIC

분류 문제에 대한 메트릭 열거.

ID	설명
CLASSIFICATION_ACCURACY	모든 클래스의 예측 정확도 측면에서 모델 품질
CLASSIFICATION_AVERAGE_PRECISION	모델 정확도 평균
CLASSIFICATION_BALANCED_ACCURACY	균형 잡힌 예측 정확도
CLASSIFICATION_F1	F1 스코어. 모델 정밀도와 재현율 간의 조화 평균
CLASSIFICATION_JACCARD	자카드 스코어
CLASSIFICATION_PRECISION	대상 클래스에 대한 트루 포지티브 예측의 모델 정확도
CLASSIFICATION_RECALL	모델 완성도
CLASSIFICATION_ROC_AUC	에러 커브 아래 영역
CLASSIFICATION_TOP_K_ACCURACY	k 예측 라벨 위에 나타나는 올바른 라벨의 빈도

ENUM_LOSS_FUNCTION

손실 함수 계산 vector::Loss을 위한 열거.

ID	설명
LOSS_MSE	평균 제곱근 오차
LOSS_MAE	평균 절대 오차
LOSS_CCE	범주형 교차 엔트로피
LOSS_BCE	이진 크로스엔트로피
LOSS_MAPE	평균 절대(Mean Absolute) 백분율 오류
LOSS_MSLE	평균 제곱 대수 오차
LOSS_KLD	쿨백-라이블러 발산
LOSS_COSINE	코사인 유사성/근접성
LOSS_POISSON	푸아송
LOSS_HINGE	힌지
LOSS_SQ_HINGE	스퀘어드 힌지
LOSS_CAT_HINGE	범주형 힌지
LOSS_LOG_COSH	하이퍼볼릭 코사인의 로그
LOSS_HUBER	후버

© 2000-2025, MetaQuotes Ltd.

ENUM_ACTIVATION_FUNCTION

활성화 함수 vector::Activation을 위한 열거와 활성화 함수 도함수 vector::Derivative을 위한 열거.

ID	설명
AF_ELU	Exponential Linear Unit
AF_EXP	지수
AF_GELU	가우시안 오차 선형 단위
AF_HARD_SIGMOID	하드 시그모이드
AF_LINEAR	선형
AF_LRELU	Leaky Rectified Linear Unit
AF_RELU	REctified Linear Unit
AF_SELU	스케일링된 지수 선형 단위
AF_SIGMOID	시그모이드
AF_SOFTMAX	소프트맥스
AF_SOFTPLUS	소프트플러스
AF_SOFTSIGN	소프트사인
AF_SWISH	스위시
AF_TANH	하이퍼볼릭 탄젠트 함수
AF_TRELU	Thresholded Rectified Linear Unit

ENUM_SORT_MODE

다음에 대한 정렬 유형 열거Sort 함수.

ID	설명
SORT_ASCENDING	오름차순 정렬
SORT_DESCENDING	내림차순 정렬

ENUM_MATRIX_AXIS

행렬에서 모든 통계 함수에서 축을 지정하기 위한 열거형

© 2000-2025, MetaQuotes Ltd.

ID	설명
AXIS_NONE	축이 지정되지 않았습니다. 마치 벡터인 것처럼 모든 행렬 요소에 대해 계산이 수행됩니다.Flast 메서드를 참고).
AXIS_HORZ	수평축
AXIS_VERT	수직축

© 2000-2025, MetaQuotes Ltd.

초기화

행렬과 벡터를 선언하고 초기화하는 방법에는 여러 가지가 있습니다.

함수	액션
Assign	자동 캐스트로 행렬, 벡터 또는 배열을 복사
CopyIndicatorBuffer	지정된 지표의 지정된 양의 버퍼를 벡터로 가져오기
CopyRates	지정된 양의 지정된 기호 기간의 MqlRates 구조의 과거 시리즈를 행렬 또는 벡터로 가져옵니다.
CopyTicks	틱을 MqlTick 구조에서 행렬 또는 벡터로 가져오기
CopyTicksRange	지정된 날짜 범위 내에서 MqlTick 구조에서 행렬 또는 벡터로 틱 가져오기
Eye	대각선에 1이 있고 이외의 곳에 0이 있는 행렬을 반환
Identity	지정된 크기의 항등 행렬을 생성
Ones	1로 채워진 새 행렬을 만들고 반환
Zeros	0으로 채워진 새 행렬을 만들고 반환
Full	주어진 값으로 채워진 새로운 행렬을 생성하고 반환
Tri	주어진 대각선 이하에 1이 있고 다른 위치에 0이 있는 행렬을 구성
Init	행렬 또는 벡터를 초기화합니다.
Fill	지정된 값으로 기존 행렬 또는 벡터 채우기
Random	정적 함수 랜덤 값으로 채워진 새로운 행렬 또는 벡터를 작성하고 반환하십시오. 램덤 값은 지정된 범위 내에서 균일하게 생성됩니다.

크기를 지정하지 않은 선언(데이터에 대한 메모리 할당 없음):

```
matrix          matrix_a;   //double 타입 행렬
matrix<double>  matrix_a1;  // double matrix을 선언하는 또 다른 방법; 템플릿에서 사용 가능
matrixf         matrix_a2;  // float matrix
matrix<float>   matrix_a3;  // float matrix
vector          vector_a;   // double vector
vector<double>  vector_a1;
vectorf         vector_a2;  // float vector
vector<float>   vector_a3;
```

지정된 크기의 선언(데이터에 대한 메모리 할당이 있지만 초기화는 없음):

```
matrix          matrix_a(128,128);          // 매개변수는 상수이거나
matrix<double>  matrix_a1(InpRows,InpCols); // 혹은 변수
matrixf         matrix_a2(1,128);           // 수평 벡터의 아날로그
matrix<float>   matrix_a3(InpRows,1);       // 수직 행렬의 아날로그
```

© 2000-2025, MetaQuotes Ltd.

```
  vector         vector_a(256);
  vector<double> vector_a1(InpSize);
  vectorf        vector_a2(SomeFunc());       // 함수 SomeFunc는 벡터 크기를 설정하는 데
  vector<float>  vector_a3(InpSize+16);       // 표현식이 매개변수로 사용될 수 있습니다.
```

초기화 선언(행렬 및 벡터 크기는 초기화 시퀀스에 의해 결정됨):

```
  matrix         matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
  matrix<double> matrix_a1=matrix_a;                    // 동일한 타입의 행렬이어야 함
  matrixf        matrix_a2={{1,0,0},{0,1,0},{0,0,1}};
  matrix<float>  matrix_a3={{1,2},{3,4}};
  vector         vector_a={-5,-4,-3,-2,-1,0,1,2,3,4,5};
  vector<double> vector_a1={1,5,2.4,3.3};
  vectorf        vector_a2={0,1,2,3};
  vector<float>  vector_a3=vector_a2;                   // 동일한 타입의 벡터이어야 함
```

초기화와 함께 선언:

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
...

  matrix  matrix_a(size_m,size_k,MatrixArange,-M_PI,0.1); // 먼저 size_m x size_k 크기의
  matrixf matrix_a1(10,20,MatrixArange);                  // 행렬을 생성한 후 기본 매개변
  vector  vector_a(size,VectorArange,-10.0);              // 벡터를 생성한 후 하나의 매개
  vectorf vector_a1(128,VectorArange);
```

데이터 메모리는 항상 동적이므로 행렬 또는 벡터의 차원을 변경할 수 있습니다.

정적 메서드

특정 방식으로 초기화된 지정된 크기의 행렬 및 벡터를 생성하기 위한 정적 메서드:

© 2000-2025, MetaQuotes Ltd.

```
matrix          matrix_a =matrix::Eye(4,5,1);
matrix<double> matrix_a1=matrix::Full(3,4,M_PI);
matrixf         matrix_a2=matrixf::Identity(5,5);
matrixf<float> matrix_a3=matrixf::Ones(5,5);
matrix          matrix_a4=matrix::Tri(4,5,-1);
vector          vector_a =vector::Ones(256);
vectorf         vector_a1=vector<float>::Zeros(16);
vector<float>   vector_a2=vectorf::Full(128,float_value);
```

이미 생성된 행렬 및 벡터를 초기화하는 메서드:

```
matrix  matrix_a;
matrix_a.Init(size_m,size_k,MatrixArange,-M_PI,0.1);
matrixf matrix_a1(3,4);
matrix_a1.Init(10,20,MatrixArange);
vector  vector_a;
vector_a.Init(128,VectorArange);
vectorf vector_a1(10);
vector_a1.Init(vector_size,VectorArange,start_value,step);

matrix_a.Fill(double_value);
vector_a1.Fill(FLT_MIN);
matrix_a1.Identity();
```

© 2000-2025, MetaQuotes Ltd.

Assign

자동 캐스트로 행렬, 벡터 또는 배열을 복사

```
bool matrix::Assign(
  const matrix<T>  &mat     // 복사된 행렬
   );
bool matrix::Assign(
  const void       &array[] // 복사된 행렬
   );
bool vector::Assign(
  const vector<T>  &vec     // 복사된 행렬
   );
bool vector::Assign(
  const void       &array[] // 복사된 행렬
   );
```

매개 변수

m, v or array

【n】값이 복사되는 행렬, 벡터 또는 배열.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환

참조

복사와는 달리, Assign 메서드는 배열 복사도 허용합니다. 이 경우 자동 캐스트가 발생하고 결과 행렬 또는 벡터가 복사된 배열의 크기에 맞게 조정됩니다.

예:

```
//--- 행렬 복사
  matrix a= {{2, 2}, {3, 3}, {4, 4}};
  matrix b=a+2;
  matrix c;
  Print("matrix a \n", a);
  Print("matrix b \n", b);
  c.Assign(b);
  Print("matrix c \n", a);

//--- 배열을 행렬에 복사
  matrix double_matrix=matrix::Full(2,10,3.14);
  Print("double_matrix before Assign() \n", double_matrix);
  int int_arr[5][5]= {{1, 2}, {3, 4}, {5, 6}};
  Print("int_arr: ");
  ArrayPrint(int_arr);
  double_matrix.Assign(int_arr);
```

© 2000-2025, MetaQuotes Ltd.

```
Print("double_matrix after Assign(int_arr) \n", double_matrix);
/*
 matrix a
 [[2,2]
  [3,3]
  [4,4]]
 matrix b
 [[4,4]
  [5,5]
  [6,6]]
 matrix c
 [[2,2]
  [3,3]
  [4,4]]

 double_matrix before Assign()
 [[3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]
  [3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14,3.14]]

 int_arr:
     [,0][,1][,2][,3][,4]
 [0,]   1   2   0   0   0
 [1,]   3   4   0   0   0
 [2,]   5   6   0   0   0
 [3,]   0   0   0   0   0
 [4,]   0   0   0   0   0

 double_matrix after Assign(int_arr)
 [[1,2,0,0,0]
  [3,4,0,0,0]
  [5,6,0,0,0]
  [0,0,0,0,0]
  [0,0,0,0,0]]

*/
```

또다른 참조

Copy

© 2000-2025, MetaQuotes Ltd.

CopyIndicatorBuffer

지정된 지표의 지정된 양의 버퍼를 벡터로 가져오기.

데이터는 벡터에 할당된 물리적 메모리의 시작 부분에서 가장 오래된 요소를 찾는 벡터에 복사됩니다. 세 가지 함수 옵션이 있습니다.

초기 위치 및 필요한 요소 수에 의한 액세스

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // 지표 핸들
   ulong     buffer_index,         // 지표 넘버 수
   ulong     start_pos,            // 복사할 시작 위치
   ulong     count                 // 복사할 요소 수
   );
```

시작일 및 필수 요소수에 따른 액세스

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // 지표 핸들
   ulong     buffer_index,         // 지표 넘버 수
   datetime  start_time,           // 복사 시작할 날짜
   ulong     count                 // 복사할 요소 수
   );
```

필요한 시간 간격의 초기 및 최종 날짜까지 액세스

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // 지표 핸들
   ulong     buffer_index,         // 지표 넘버 수
   datetime  start_time,           // 복사 시작할 날짜
   datetime  stop_time             // 언제까지 복사할 날짜
   );
```

패러미터

indicator_handle

[in] 관련 지표 함수에서 가져온 지표 핸들.

buffer_index

[in] 지표 버퍼 수

start_pos

[in] 첫 번째 복사된 요소 인덱스의 수.

count

[in] 복사된 요소 수.

start_time

【n】첫 번째 요소에 해당하는 바의 시간.

stop_time

【n】마지막 요소에 해당하는 바의 시간.

© 2000-2025, MetaQuotes Ltd.

반환 값

이 함수는 성공하면 'true'를 반환하고 오류 발생하면 'false'를 반환합니다.

주의

복사된 데이터의 요소(인덱스 buffer_index가 있는 지표 버퍼)는 현재부터 과거까지 카운트 다운되므로 시작 위치가 0인 것은 현재 바(현재 바에 대한 지표 값)를 의미합니다.

알 수 없는 양의 데이터를 복사할 때는 CopyBuffer() 함수가 수신 벡터의 크기를 복사된 데이터의 크기로 분배하려고 하기 때문에 크기를 지정하지 않고(데이터에 대한 메모리 할당 없이) 벡터를 선언해야 합니다.

지표 값을 부분적으로 복사하려면 필요한 양만큼 복사하는 중간 벡터를 사용해야 합니다. 이 중간 벡터에서 멤버별로 필요한 수만큼의 값을 수신 벡터의 필요한 위치에 복사할 수 있습니다.

미리 정해진 양의 데이터를 복사하는 경우 다음을 권장합니다. 불필요한 메모리 재 할당을 피하기 위해 벡터를 미리 선언하고 크기를 지정하십시오.

지표에서 데이터를 요청할 때 요청된 시계열이 아직 구성되지 않았거나 로드/구성을 시작하는 동안 서버에서 다운로드해야 하는 경우 함수는 즉시 false를 반환합니다.

EA 또는 스크립트에서 데이터를 요청할 때, 서버에서 다운로드는 터미널에 로컬에 적절한 데이터가 없는 경우 시작되거나 로컬 기록에서 데이터를 구성할 수 있지만 필요한 시간 프레임이 아직 준비되지 않은 경우 필요한 시계열 구성이 시작됩니다. 이 함수는 제한 시간이 만료될 때까지 사용 가능한 양을 반환합니다.

더 참조할 만한 내용

CopyBuffer

© 2000-2025, MetaQuotes Ltd.

CopyRates

MqlRate의 지정된 양의 지정된 기호 기간의 구조를 행렬 또는 벡터로 변환한 것의 역사적인 시리즈를 가져옵니다.. 요소는 현재부터 과거까지 카운트다운됩니다. 즉 시작 위치가 0이면 현재 바를 의미합니다.

가장 오래된 요소가 행렬/벡터의 시작 부분에 배치되도록 데이터가 복사됩니다. 세 가지 함수 옵션이 있습니다.

초기의 포지션 및 필요한 요소 수에 의한 액세스

```
bool  matrix::CopyRates(
   string           symbol,       // 심볼명
   ENUM_TIMEFRAMES  period,       // 주기
   ulong            rates_mask,   // 요청된 시리즈를 지정하기 위한 플래그의 조합
   ulong            start,        // ~부터 복사가 시작되는 최초 바의 인데스
   ulong            count         // 얼마나 복사할지
   );
```

초기 날짜 및 필수 요소 수에 의한 액세스

```
bool  matrix::CopyRates(
   string           symbol,       // 심볼명
   ENUM_TIMEFRAMES  period,       // 주기
   ulong            rates_mask,   // 요청된 시리즈를 지정하기 위한 플래그의 조합
   datetime         from,         // 시작 날짜
   ulong            count         // 얼마나 복사할지
   );
```

요청되는 시간 간격의 초기 및 최종 날짜까지 액세스

```
bool  matrix::CopyRates(
   string           symbol,       // 심볼명
   ENUM_TIMEFRAMES  period,       // 주기
   ulong            rates_mask,   // 요청된 시리즈를 지정하기 위한 플래그의 조합
   datetime         from,         // 시작 날짜
   datetime         to            // 끝나는 날짜
   );
```

벡터 방법

초기의 포지션 및 필요한 요소 수에 의한 액세스

```
bool  vector::CopyRates(
   string           symbol,       // 심볼명
   ENUM_TIMEFRAMES  period,       // 주기
   ulong            rates_mask,   // 요청된 시리즈를 지정하기 위한 플래그의 조합
   ulong            start,        // ~부터 복사가 시작되는 최초 바의 인데스
   ulong            count         // 얼마나 복사할지
   );
```

초기 날짜 및 필수 요소 수에 의한 액세스

© 2000-2025, MetaQuotes Ltd.

```
bool  vector::CopyRates(
   string           symbol,       // 심볼명
   ENUM_TIMEFRAMES  period,       // 주기
   ulong            rates_mask,   // 요청된 시리즈를 지정하기 위한 플래그의 조합
   datetime         from,         // 시작 날짜
   ulong            count         // 얼마나 복사할지
   );
```

요청되는 시간 간격의 초기 및 최종 날짜까지 액세스

```
bool  vector::CopyRates(
   string           symbol,       // 심볼명
   ENUM_TIMEFRAMES  period,       // 주기
   ulong            rates_mask,   // 요청된 시리즈를 지정하기 위한 플래그의 조합
   datetime         from,         // 시작 날짜
   datetime         to            // 끝나는 날짜
   );
```

매개 변수

Symbol

[in] 심볼

period

[in] 주기.

rates_mask

【n】요청된 시리즈의 유형을 지정하는 플래그의 ENUM_COPY_RATES 열거형 조합입니다. 벡터로 복사할 때 ENUM_COPY_RATES 열거형에서 하나의 값만 지정할 수 있으며 그렇지 않으면 오류가 발생합니다.

start

[in]처음 복사된 요소 인덱스.

count

[in】복사된 요소의 수

from

【n】첫 번째 요소에 해당하는 바 시간입니다.

to

【n】마지막 요소에 해당하는 막대 시간입니다.

반환값

성공하면 true를 반환하고 그렇지 않고 오류.이면 false를 반환합니다.

참조

요청된 데이터의 간격이 서버에서 사용 가능한 데이터를 완전히 벗어나면 함수는 다음을 반환합니다 .false. 데이터가 외부에 있는 경우TERMINAL_MAXBARS (차트의 최대 바의 수)가 요청되면 함수도 다음을 반환합니다.false.

© 2000-2025, MetaQuotes Ltd.

EA 또는 스크립트에서 데이터를 요청할 때,서버에서 다운로드는 터미널에 로컬에 적절한 데이터가 없는 경우 시작되거나 로컬 기록에서 데이터를 구성할 수 있지만 아직 준비되지 않은 경우 필요한 시계열 구성이 시작됩니다. 함수는 제한 시간이 만료될 때까지 준비될 양을 반환하지만 기록 다운로드는 계속되고 함수는 이후 유사한 요청 중에 더 많은 데이터를 반환합니다.

시작 날짜와 필요한 항목 수로 데이터를 요청할 경우 (전에) 지정된 날짜보다 이전 또는 같은 날짜의 데이터만 반환됩니다. 인터벌이 설정되고 최대 1초로 간주됩니다. 즉, 값이 반환되는 바(볼륨, 스프레드, 시가, 고가, 저가, 종가 또는 시간)의 개장일은 항상 지정된 날짜보다 작거나 같습니다.

지정된 날짜 범위의 데이터를 요청할 때 요청된 인터벌에 해당하는 데이터만 반환됩니다. 인터벌이 설정되고 최대 1초로 간주됩니다. 즉, 값이 반환되는 바(볼륨, 스프레드, 인디케이터 버퍼 값, 시가, 고가, 저가, 종가 또는 시간)의 시작 시간은 항상 요청된 인터벌에 있습니다.

예를 들어 현재 요일이 토요일인 경우 설정을 통해 주간 시간대에 데이터 복사를 시도하면 함수는 0을 반환합니다.*start_time=마지막_화요일* 그리고*stop_time=Last_Friday* 주간 시간대의 개장 시간은 항상 일요일이지만 지정된 범위에 속하는 주간 바는 하나도 없기 때문입니다.

현재 완성되지 바에 해당하는 값을 가져와야 하는 경우 다음을 나타내는 첫 번째 호출 형식을 사용할 수 있습니다. *start_pos*=0 및*count*=1.

ENUM_COPY_RATES

ENUM_COPY_RATES 열거형에는 행렬 또는 배열에 전달할 데이터 유형을 지정하는 플래그가 포함되어 있습니다. 플래그 조합을 사용하면 하나의 요청으로 히스토리에서 여러 시리즈를 가져올 수 있습니다. 행렬의 행 순서는 ENUM_COPY_RATES 열거형의 값 순서와 일치합니다. 즉, 높은 데이터가 있는 행은 낮은 데이터가 있는 행보다 매트릭스에서 항상 더 높습니다.

ID	Value	설명
COPY_RATES_OPEN	1	시가 시리즈
COPY_RATES_HIGH	2	고가 시리즈
COPY_RATES_LOW	4	저가 시리즈
COPY_RATES_CLOSE	8	종가 시리즈
COPY_RATES_TIME	16	타임 시리즈 (바 오픈 시간) 벡터와 행렬(vectord 및 matrixf)의 float에서 시간을 얻으면 ~100초의 손실을 초래합니다. 왜냐하면 float 정확도는 상당히 제한되며 1<<24보다 큰 정수는 float로 정확하게 표현할 수 없기 때문입니다.
COPY_RATES_VOLUME_TICK	32	틱 볼륨
COPY_RATES_VOLUME_REAL	64	트레이드 볼륨
COPY_RATES_SPREAD	128	스프레드
조합		
COPY_RATES_OHLC	15	시가, 고가, 저가, 종가 시리즈

© 2000-2025, MetaQuotes Ltd.

ID	Value	설명
COPY_RATES_OHLCT	31	시가, 고가, 저가, 종가 시리즈
데이터 정렬		
COPY_RATES_VERTICAL	32768	계열은 세로 축을 따라 행렬에 복사됩니다. 수신된 계열 값은 행렬에서 수직으로 정렬됩니다. 즉, 가장 오래된 데이터가 첫 번째 행에 있고 가장 최근 데이터가 마지막 행렬 행에 있습니다. 기본 복사를 사용하면 시리즈가 가로 축을 따라 행렬에 추가됩니다. 플래그는 행렬에 복사할 때만 적용 가능합니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//---행렬에 쿼트 얻기
  matrix matrix_rates;
  if(matrix_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLCT, 1, 10))
    Print("matrix rates: \n", matrix_rates);
  else
    Print("matrix_rates.CopyRates failed. Error ", GetLastError());
//--- 확인
  MqlRates mql_rates[];
  if(CopyRates(Symbol(), PERIOD_CURRENT, 1, 10, mql_rates)>0)
   {
    Print("mql_rates array:");
    ArrayPrint(mql_rates);
   }
  else
    Print("CopyRates(Symbol(), PERIOD_CURRENT,1, 10, mql_rates). Error ", GetLastErro
//--- 벡터로 쿼트 가져오기 = 잘못된 호출
  vector vector_rates;
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_OHLC, 1, 15))
    Print("vector_rates COPY_RATES_OHLC: \n", vector_rates);
  else
    Print("vector_rates.CopyRates COPY_RATES_OHLC failed. Error ", GetLastError());
//--- 종가를 벡터로 가져옵니다.
  if(vector_rates.CopyRates(Symbol(), PERIOD_CURRENT, COPY_RATES_CLOSE, 1, 15))
    Print("vector_rates COPY_RATES_CLOSE: \n", vector_rates);
  else
    Print("vector_rates.CopyRates failed. Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
 };
/*
   matrix rates:
   [[0.99686,0.99638,0.99588,0.99441,0.99464,0.99594,0.99698,0.99758,0.99581,0.995280
    [0.99708,0.99643,0.99591,0.9955000000000001,0.99652,0.99795,0.99865,0.99764,0.996
    [0.9961100000000001,0.99491,0.99426,0.99441,0.99448,0.99494,0.9964499999999999,0.9
    [0.99641,0.99588,0.99441,0.99464,0.99594,0.99697,0.99758,0.99581,0.995280000000000
    [1662436800,1662440400,1662444000,1662447600,1662451200,1662454800,1662458400,1662
   mql_rates array:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2022.09.06 04:00:00 0.99686 0.99708 0.99611 0.99641          4463        0
   [1] 2022.09.06 05:00:00 0.99638 0.99643 0.99491 0.99588          4519        0
   [2] 2022.09.06 06:00:00 0.99588 0.99591 0.99426 0.99441          3060        0
   [3] 2022.09.06 07:00:00 0.99441 0.99550 0.99441 0.99464          3867        0
   [4] 2022.09.06 08:00:00 0.99464 0.99652 0.99448 0.99594          5280        0
   [5] 2022.09.06 09:00:00 0.99594 0.99795 0.99494 0.99697          7227        0
   [6] 2022.09.06 10:00:00 0.99698 0.99865 0.99645 0.99758         10130        0
   [7] 2022.09.06 11:00:00 0.99758 0.99764 0.99472 0.99581          7012        0
   [8] 2022.09.06 12:00:00 0.99581 0.99604 0.99360 0.99528          6166        0
   [9] 2022.09.06 13:00:00 0.99528 0.99570 0.99220 0.99259          6950        0
   vector_rates.CopyRates COPY_RATES_OHLC failed. Error 4003
   vector_rates COPY_RATES_CLOSE:
   [0.9931,0.99293,0.99417,0.99504,0.9968399999999999,0.99641,0.99588,0.99441,0.99464,
*/
```

또다른 참조

시계열 및 지표에 대한 액세스,CopyRates

© 2000-2025, MetaQuotes Ltd.

CopyTicks

틱을 MqlTick 구조에서 행렬 또는 벡터로 가져옵니다. 요소는 과거부터 현재까지 계산됩니다. 즉, 인덱스가 0인 틱이 가장 오래된 것입니다. MqlTick 구조에서 틱을 행렬 또는 벡터로 가져옵니다. 틱을 분석하려면 다음을 확인하십시오. *flags*는 틱에서 정확히 무엇이 변경되었는지 보여주는 필드로 이 부분을 확인합니다.

```
bool  matrix::CopyTicks(
   string           symbol,              // 심볼명
   ulong            flags,               // 수신되어야 할 틱의 타입을 알려주는 flag
   ulong            from_msc,            // 틱이 요청된 시간부터
   ulong            count                // 수신되어야 할 틱의 수
   );
```

벡터 Method

```
bool  vector::CopyTicks(
   string           symbol,              // 심볼명
   ulong            flags,               // 수신되어야 할 틱의 타입을 알려주는 flag
   ulong            from_msc,            // 틱이 요청된 시간부터
   ulong            count                // 수신되어야 할 틱의 수
   );
```

매개 변수

Symbol

[in] 심볼

flags

[in] 요청된 데이터의 내용을 나타내는 ENUM_COPY_TICKS 열거의 플래그 조합입니다.벡터로 복사할 때 ENUM_COPY_TICKS 열거형에서 하나의 값만 지정할 수 있습니다. 그렇지 않으면 오류가 발생합니다.

from_msc

[in] 틱이 요청되는 때부터 시작 시간. 시간은 1970년 1월 1일 이후부터 밀리초 단위로 지정됩니다. from_msc=0인 경우 'count'와 동일한 마지막 틱 수가 반환됩니다.

count

[in] 요청된 틱의 수 매개변수 'from_msc' 및 'count'가 지정되지 않은 경우 사용 가능한 모든 틱이 기록되지만 2000개 이하입니다.

반환값

성공 시 true를 반환하고 오류가 발생하면 false를 반환합니다.

참조

CopyTicks()의 첫 번째 호출때 하드 드라이브에 저장된 관련 심볼의 틱 데이터베이스와 동기화를 시작합니다. 로컬 데이터베이스가 요청된 모든 틱을 제공하지 않는 경우 누락된 틱은 거래 서버에서 자동으로 다운로드됩니다. CopyTicks()에 지정된 ***from_msc***에서 현재까지의 틱이 동기화됩니다.. 그 후, 이 심볼에 대해 도착하는 모든 틱은 틱 데이터베이스에 추가되어 동기화된 상태로 유지됩니다.

© 2000-2025, MetaQuotes Ltd.

from_msc 및 *count* 매개변수가 지정되지 않은 경우 사용 가능한 모든 틱(2000 이하)은 행렬/벡터에 기록됩니다.

지표에서 CopyTicks() 메서드는 결과를 즉시 반환합니다: CopyTick()이 지표에서 호출되면 CopyTick()은 심볼의 사용 가능한 모든 틱을 즉시 반환하고 사용 가능한 데이터가 충분하지 않은 경우 틱 데이터베이스의 동기화를 시작합니다. 동일한 심볼에 있는 모든 지표는 하나의 공통 스레드에서 작동하므로 지표는 동기화의 완료를 기다릴 수 없습니다. 동기화 후 CopyTicks()는 다음 호출 중에 요청된 모든 틱을 반환합니다. 지표에서 OnCalculate() 함수는 각 틱이 도착한 후에 호출됩니다.

Expert Advisors 및 스크립트에서 CopyTicks()는 45초 동안 결과를 기다릴 수 있습니다. 지표와는 달리 모든 Expert Advisor 또는 스크립트는 별도의 스레드에서 작동하므로 동기화가 완료되기까지 45초 정도를 기다릴 수 있습니다. 필요한 양의 틱이 이 시간 동안 동기화되지 않으면 CopyTicks()는 시간 초과까지의 사용 가능한 틱을 반환하고 동기화를 계속합니다. Expert Advisors의 OnTick()은 시장에서 일어난 변화에 대해서만 Expert Advisor에게 알립니다. 모든 틱을 처리하는 핸들러가 아닙니다. 이것은 일괄적인 변경일 수 있습니다: 터미널은 동시에 여러 틱을 수신할 수 있지만 OnTick()은 Expert Advisor에 최신 시장 상태를 알리기 위해 한 번만 호출되어 집니다.

Rate of data return: 터미널은 빠른 액세스 캐시에 각 상품에 대한 4096개의 마지막 틱을 저장합니다(시장 심도가 실행 중인 심볼의 경우 65536틱). 이 데이터와 관련한 요청이 가장 빠르게 실행됩니다. 현재 거래 세션에 대해 요청된 틱이 캐시의 외부에 있는 경우 CopyTicks()는 터미널 메모리에 저장된 틱을 호출합니다. 이러한 요청은 완료되는 데 더 많은 시간이 필요합니다. 이 경우 드라이브에서 데이터를 읽습니다. 그러므로 가장 느린 요청은 다른 날의 틱을 요청하는 요청입니다.

ENUM_COPY_TICKS

ENUM_COPY_TICKS 열거형에는 행렬 또는 배열에 전달할 데이터 유형을 지정하는 플래그가 포함되어 있습니다. 플래그 조합을 사용하면 하나의 요청으로 히스토리에서 여러 시리즈를 가져올 수 있습니다. 행렬의 행 순서는 ENUM_COPY_TICKS 열거형의 값 순서와 일치합니다. 즉, 높은 데이터가 있는 행은 낮은 데이터가 있는 행보다 매트릭스에서 항상 더 높습니다.

ID	Value	설명
COPY_TICKS_INFO	1	입찰 및/또는 요청 가격 변동을 포함하는 틱
COPY_TICKS_TRADE	2	Last 및/또는 Volume 가격 변동을 포함하는 틱
COPY_TICKS_ALL	3	모든 틱
COPY_TICKS_TIME_MS	1<<8	Tick time in milliseconds
COPY_TICKS_BID	1<<9	Bid price
COPY_TICKS_ASK	1<<10	Ask price
COPY_TICKS_LAST	1<<11	Last price (last deal price)
COPY_TICKS_VOLUME	1<<12	Last price Volume
COPY_TICKS_FLAGS	1<<13	Tick Flags
데이터 정렬		
COPY_TICKS_VERTICAL	32768	틱은 수직 축을 따라 행렬에 복사됩니다. 수신된 틱은 행렬에서 수직으로 배열됩니다. 즉, 가장 오래된 틱은 첫 번째 행에 있고 가장 최근 틱은 마지막 행렬 행에 있습니다.

© 2000-2025, MetaQuotes Ltd.

ID	Value	설명
		디폴트 복사를 사용하면 눈금이 가로 축을 따라 행렬에 추가됩니다. 플래그는 행렬에 복사할 때만 적용 가능합니다.

틱 플래그를 분석하여 어떤 데이터가 변경되었는지 확인합니다:

- TICK_FLAG_BID — 틱이 bid 가격을 변경했습니다.
- TICK_FLAG_ASK — 틱이 ask 가격을 변경했습니다.
- TICK_FLAG_LAST — 틱이 마지막 거래 가격을 변경했습니다.
- TICK_FLAG_VOLUME — 틱이 볼륨을 변경했습니다.
- TICK_FLAG_BUY — 틱은 매수 거래의 결과입니다.
- TICK_FLAG_SELL — 틱은 매도 거래의 결과입니다.

또다른 참조

시계열 및 지표에 대한 액세스,CopyTicks

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange

틱을 MqlTick 구조에서 지정된 데이터 범위 내에서 행렬 또는 벡터로 가져옵니다. 요소는 과거부터 현재까지 계산됩니다. 즉, 인덱스가 0인 틱이 가장 오래된 것입니다. 틱을 분석하려면 다음을 확인하십시오. *flags*는 틱에서 정확히 무엇이 변경되었는지 보여주는 필드로 이 부분을 확인합니다.

```
bool  matrix::CopyTicksRange(
   string           symbol,                 // 심볼명
   ulong            flags,                  // 수신되어야 할 틱의 타입을 알려주는 flag
   ulong            from_msc,               // 틱이 요청된 시간부터
   ulong            to_msc                  // 틱이 요청되기까지 시간
   );
```

벡터 Method

```
bool  vector::CopyTicksRange(
   string           symbol,                 // 심볼명
   ulong            flags,                  // 수신되어야 할 틱의 타입을 알려주는 flag
   ulong            from_msc,               // 틱이 요청된 시간부터
   ulong            to_msc                  // 틱이 요청되기까지 시간
   );
```

매개 변수

Symbol

[in] 심볼

flags

[in] 요청된 데이터의 내용을 나타내는 ENUM_COPY_TICKS 열거의 플래그 조합입니다.벡터로 복사할 때 ENUM_COPY_TICKS 열거형에서 하나의 값만 지정할 수 있습니다. 그렇지 않으면 오류가 발생합니다.

from_msc

[in] 틱이 요청되는 때부터 시작 시간. 시간은 1970년 1월 1일 이후부터 밀리초 단위로 지정됩니다. 'from_msc' 매개변수를 지정하지 않으면 히스토리의 처음부터 틱이 반환됩니다. Ticks with the time >= from_msc가 반환 됨.

to_msc

[in] Time up to which ticks are requested. 시간은 1970년 1월 1일 이후부터 밀리초 단위로 지정됩니다. Ticks with the time <= to_msc가 반환됨. to_msc 매개변수를 지정하지 않으면 히스토리 끝까지의 모든 틱이 반환됩니다.

반환값

성공하면 true를 반환하고 오류가 발생하면 false를 반환합니다. GetLastError()는 다음과 같은 오류를 반환할 수 있습니다.

- ERR_HISTORY_TIMEOUT — 틱 동기화 시간 초과가 만료되었습니다. 함수가 가진 모든 것을 반환했습니다.
- ERR_HISTORY_SMALL_BUFFER — 정적 버퍼가 너무 작습니다. 배열이 저장할 수 있는 양만 반환되었습니다.
- ERR_NOT_ENOUGH_MEMORY — 지정된 범위의 기록 데이터를 동적 틱 배열로 수신하기 위한 메모리가 충분하지 않습니다. 틱 배열에 충분한 메모리를 할당하지 못했습니다.

© 2000-2025, MetaQuotes Ltd.

틱 플래그를 분석하여 어떤 데이터가 변경되었는지 확인합니다:

- TICK_FLAG_BID — 틱이 bid 가격을 변경했습니다.
- TICK_FLAG_ASK — 틱이 ask 가격을 변경했습니다.
- TICK_FLAG_LAST — 틱이 마지막 거래 가격을 변경했습니다.
- TICK_FLAG_VOLUME — 틱이 볼륨을 변경했습니다.
- TICK_FLAG_BUY — 틱은 매수 거래의 결과입니다.
- TICK_FLAG_SELL — 틱은 매도 거래의 결과입니다.

참조

CopyTicksRange() 메서드는 지정된 범위 내에서 틱을 요청하는 데 사용됩니다. 예를 들어 히스토리의 특정 날짜에 대한 틱입니다. CopyTicks()는 시작 날짜만 지정할 수 있습니다. 예를 들어 월초부터 현재까지의 모든 틱을 수신하기 위한 경우등입니다.

또다른 참조

시계열 및 지표에 대한 액세스,CopyTicksRang

© 2000-2025, MetaQuotes Ltd.

Eye

통계 함수. 주대각선이 1이고 다른 곳이 0인 지정된 크기의 행렬을 구성합니다. 대각선에 1이 있고 다른 곳에 0이 있는 행렬을 반환합니다.

```
static matrix matrix::Eye(
  const ulong  rows,         // 행의 수
  const ulong  cols,         // 열의 수
  const int    ndiag=0       // 대각선의 인덱스
   );
```

매개 변수

rows

【n】출력에서 행의 수입니다.

cols

[in] 출력에서 열의 수입니다.

ndiag=0

[in] 대각선 인덱스: 0(기본값)은 주 대각선을 나타내고 양수 값은 위쪽 대각선을, 음수 값은 아래쪽 대각선을 나타냅니다.

반환값

값이 1인 k번째 대각선을 제외하고 모든 요소가 0인 행렬입니다.

MQL5 예:

```
  matrix eye=matrix::Eye(3, 3);
  Print("eye = \n", eye);

  eye=matrix::Eye(4, 4,1);
  Print("eye = \n", eye);
  /*
   eye =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
   eye =
   [[0,1,0,0]
    [0,0,1,0]
    [0,0,0,1]
    [0,0,0,0]]
  */
```

파이썬 예:

```
np.eye(3, dtype=int)
```

© 2000-2025, MetaQuotes Ltd.

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])


 np.eye(4, k=1)
array([[0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 0.]])
```

© 2000-2025, MetaQuotes Ltd.

Identity

지정된 크기(반드시 정사각형일 필요는 없음)의 항등 행렬을 생성하는 정적 함수입니다. 항등 행렬은 주 대각선에 1을 포함하고 다른 곳에는 0을 포함합니다. 주대각선은 [0,0],[1,1],[2,2] 등과 같이 동일한 행 및 열 인덱스를 갖는 행렬 요소로 구성됩니다. 새 항등 행렬을 만듭니다.

이미 존재하는 행렬을 항등행렬로 변환하는 메서드인 Identity도 있습니다.

```
static matrix matrix::Identity(
  const ulong  rows,         // 행의 수
  const ulong  cols,         // 열의 수
   );

void matrix::Identity();
```

매개 변수

rows

【n】 nxn 행렬의 행(및 열) 수입니다.

반환값

항등 행렬을 반환합니다. 항등 행렬은 주 대각선에 1이 있는 정사각 행렬입니다.

MQL5 예:

```
  matrix identity=matrix::Identity(3,3);
  Print("identity = \n", identity);
/*
   identity =
   [[1,0,0]
    [0,1,0]
    [0,0,1]]
*/
  matrix identity2(3,5);
  identity2.Identity();
  Print("identity2 = \n", identity2);
/*
   identity2 =
   [[1,0,0,0,0]
    [0,1,0,0,0]
    [0,0,1,0,0]]
*/
```

파이썬 예:

```
np.identity(3)
array([[1.,  0.,  0.],
       [0.,  1.,  0.],
```

© 2000-2025, MetaQuotes Ltd.

```
        [0.,  0.,  1.]])
```

© 2000-2025, MetaQuotes Ltd.

Ones

1로 채워진 새 행렬을 만들고 반환하는 정적 함수입니다.

```
static matrix matrix::Ones(
  const ulong  rows,     // 행의 수
  const ulong  cols      // 열의 수
   );

static vector vector::Ones(
  const ulong  size,     // 벡터 사이즈
   );
```

매개 변수

rows

[in] 행 수

cols

[in] 열 수

반환값

1로 채워진 주어진 행과 열의 새로운 행렬.

MQL5 예:

```
  matrix ones=matrix::Ones(4, 4);
  Print("ones = \n", ones);
/*
ones =
   [[1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]
    [1,1,1,1]]
*/
```

파이썬 예:

```
np.ones((4, 1))
array([[1.],
       [1.]])
```

© 2000-2025, MetaQuotes Ltd.

Zeros

0으로 채워진 새 행렬을 만들고 반환하는 정적 함수입니다.

```
static matrix matrix::Zeros(
  const ulong  rows,     // 행의 수
  const ulong  cols      // 열의 수
   );

static vector vector::Zeros(
  const ulong  size,     // 벡터 사이즈
   );
```

매개 변수

rows

[in] 행 수

cols

[in] 열 수

반환값

0으로 채워진 지정된 행과 열의 새 행렬입니다.

MQL5 예:

```
  matrix zeros=matrix::Zeros(3, 4);
  Print("zeros = \n", zeros);
/*
zeros =
   [[0,0,0,0]
    [0,0,0,0]
    [0,0,0,0]]
*/
```

파이썬 예:

```
np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

© 2000-2025, MetaQuotes Ltd.

Full

정적 함수는 주어진 값으로 채워진 새로운 행렬을 생성하고 반환합니다.

```
static matrix matrix::Full(
  const ulong   rows,      // 행의 수
  const ulong   cols,      // 열의 수
  const double  value      // 채울 값
   );

static vector vector::Full(
  const ulong   size,      // vector size
  const double  value      // 채울 값
   );
```

매개 변수

rows

[in] 행 수

cols

[in] 열 수

value

【n】모든 행렬 요소를 채우는 값입니다.

반환값

지정된 값으로 채워진 주어진 행과 열의 새 행렬을 반환합니다.

MQL5 예:

```
  matrix full=matrix::Full(3,4,10);
  Print("full = \n", full);
/*
full =
   [[10,10,10,10]
    [10,10,10,10]
    [10,10,10,10]]
*/
```

예

```
np.full((2, 2), 10)
array([[10, 10],
       [10, 10]])
```

© 2000-2025, MetaQuotes Ltd.

Tri

정적 함수입니다. 주어진 대각선 이하에서 1을 사용하고 다른 곳에서는 0으로 행렬을 구성합니다.

```
static matrix matrix::Tri(
  const ulong rows,         // 행의 수
  const ulong cols,         // 열의 수
  const int   ndiag=0       // 대각선의 수
   );
```

매개 변수

rows

【n】배열에서 행의 수입니다.

cols

[in] 배열에서 열의 수입니다.

ndiag=0

【n】배열이 채워지는 종속 대각선입니다. k = 0은 주대각선이고 k < 0은 그 아래에 있고 k > 0은 위에 있습니다. 기본값은 0입니다.

반환값

아래쪽 삼각형이 1로 채워지고 다른 곳은 0으로 채워진 배열입니다.

MQL5 예:

```
   matrix matrix_a=matrix::Tri(3,4,1);
   Print("Tri(3,4,1)\n",matrix_a);
   matrix_a=matrix::Tri(4,3,-1);
   Print("Tri(4,3,-1)\n",matrix_a);

/*
   Tri(3,4,1)
   [[1,1,0,0]
    [1,1,1,0]
    [1,1,1,1]]
   Tri(4,3,-1)
   [[0,0,0]
    [1,0,0]
    [1,1,0]
    [1,1,1]]
*/
```

예

```
np.tri(3, 5, 2, dtype=int)
array([[1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1]])
```

© 2000-2025, MetaQuotes Ltd.

Init

행렬 또는 벡터를 초기화합니다.

```
void matrix::Init(
  const ulong  rows,            // 행 수
  const ulong  cols,            // 열 수
func_nameinit_func=NULL,        // 일부 범위 또는 클래스의 정적 메서드에 배치된 초기화 함수
   ...         parameters
   );

void vector::Init(
  const ulong  size,            // 벡터 크기
func_nameinit_func=NULL,        // 일부 범위 또는 클래스의 정적 메서드에 배치된 초기화 함수
   ...         parameters
   );
```

매개 변수

rows

[in] 행 수

cols

[in] 열 수

func_name

[in] 초기화 함수.

...

[in] 초기화 함수위 매개변수.

반환값

No return value.

예

```
template<typename T>
void MatrixArange(matrix<T> &mat,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<mat.Rows(); i++)
     {
      for(ulong j=0; j<mat.Cols(); j++,value+=step)
         mat[i][j]=value;
     }
  }
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
  int size_m=3, size_k=4;
  matrix  m(size_m,size_k,MatrixArange,-2.,0.1); // 먼저 size_m x size_k 크기의 초기화되
  Print("matrix m \n",m);                        // 그런 다음 초기화 시 나열된 매개변수가
  matrixf m_float(5,5,MatrixArange,-2.f,0.1f);   // float 유형의 행렬이 생성된 후 Matrix/
  Print("matrix m_float \n",m_float);
  vector  v(size_k,VectorArange,-10.0);          // 벡터가 생성된 후 매개변수가 하나인 Vec
  Print("vector v \n",v);
  /*
   matrix m
   [[-2,-1.9,-1.8,-1.7]
    [-1.6,-1.5,-1.399999999999999,-1.299999999999999]
    [-1.199999999999999,-1.099999999999999,-0.9999999999999992,-0.8999999999999992]]
   matrix m_float
   [[-2,-1.9,-1.8,-1.6999999,-1.5999999]
    [-1.4999999,-1.3999999,-1.2999998,-1.1999998,-1.0999998]
    [-0.99999976,-0.89999974,-0.79999971,-0.69999969,-0.59999967]
    [-0.49999967,-0.39999968,-0.29999968,-0.19999969,-0.099999689]
    [3.1292439e-07,0.10000031,0.20000032,0.30000031,0.4000003]]
   vector v
   [-10,-9,-8,-7]
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

Fill

기존 행렬 또는 벡터를 지정된 값으로 채웁니다.

```
void matrix::Fill(
  const double  value      // 채울 값
   );

void vector::Fill(
  const double  value      // 채울 값
   );
```

매개 변수

value

【n】모든 행렬 요소를 채우는 값입니다.

반환값

No return value. 매트릭스는 지정된 값으로 채워집니다.

예

```
   matrix matrix_a(2,2);
   matrix_a.Fill(10);
   Print("matrix_a\n",matrix_a);

/*
  matrix_a
  [[10,10]
   [10,10]]
*/
```

© 2000-2025, MetaQuotes Ltd.

랜덤

정적 함수 랜덤 값으로 채워진 새로운 행렬 또는 벡터를 작성하고 반환하십시오. 램덤 값은 지정된 범위 내에서 균일하게 생성됩니다.

```
static vector vector::Random(
  const ulong   size,       // 벡터 길이
  const double  min=0.0,    // 최소 값
  const double  max=1.0     // 최대 값
   );

static matrix matrix::Random(
  const ulong   rows,       // 행 수
  const ulong   cols        // 열 수
  const float   min=0.0,    // 최소 값
  const float   max=1.0     // 최대 값
   );
```

함수는 기존 행렬 또는 벡터를 랜덤 값으로 채웁니다. 랜덤 값은 지정된 범위 내에서 균일하게 생성됩니다.

```
void  vector::Random(
  const double  min=0.0,    // 최소 값
  const double  max=1.0     // 최대 값
   );

void  matrix::Random(
  const float  min=0.0,     // 최소 값
  const float  max=1.0      // 최대 값
   );
```

매개 변수

rows
  [in] 행 수.

cols
  [in] 열 수.

size
  [in] 벡터 길이.

min=0.0
  [in] 생성된 랜덤 넘버 샘플 중 최소값.

max=1.0
  [in] 생성된 랜덤 넘버 샘플 중 최소값.

반환 값

랜덤 값으로 채워진 주어진 행과 열의 새로운 행렬을 반환.

© 2000-2025, MetaQuotes Ltd.

예

```
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
void OnStart()
  {
   vector v1= vector::Random(6,0,4);
   Print("vector v1 \n", v1);
   /*
   vector v1
   [3.340834286841758,0.9073984578501895,3.15598511117417,0.7106475841045956,2.294010(
   */

   vector v2(4);
   v2.Random(-3,3);
   Print("vector v2 \n", v2);
   /*
   vector v2
   [-2.642032080246609,1.008521607147651,1.340623848219547,-2.789016363161853]
   */

   matrix m1 = matrix::Random(5, 5, -1, 3);
   Print("matrix m1 \n", m1);
   /*
   matrix m1
   [[1.468433369525809,1.199383023707052,0.6154667729547869,2.830012157519816,-0.06995
    [2.638591175814763,2.966658948912904,2.001837257994767,-0.0857080246987073l,0.8260
    [0.982665140643405,0.5825661973987162,-0.6615881071569371,0.4069533109215444,-0.16
    [-0.5791836119208447,-0.9332020478167575,2.801898707589577,-0.09887318478405316,-0
    [-0.1069569017521471,-0.04576879345014895,0.6320100840255642,-0.3805729270401813,0
   */

   matrix m2(3, 3);
   m2.Random(-2, 2);
   Print("matrix m2 \n", m2);
   /*
   matrix m2
   [[-1.928191291151613,-1.222873445419332,-1.965333905062949]
    [-1.077723600686929,1.053317863273741,-0.1786824246353196]
    [-1.965241401448204,1.620956906029246,-1.696386492740453]]
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

행렬 및 벡터 조작

다음은 기본 행렬 작업을 위한 메서드입니다: 채우기, 복사, 행렬의 일부 가져오기, 전치, 분할 및 정렬.

행렬의 행과 열을 사용하여 연산하는 방법도 여러 가지가 있습니다.

함수	액션
HasNan	행렬/벡터에서 NaN 수를 반환
Transpose	행렬의 축을 뒤집거나 치환합니다; 수정된 행렬을 반환합니다.
TriL	k번째 대각선 위의 요소가 0인 행렬의 복사본을 반환 Lower triangular matrix
TriU	k번째 대각선 아래의 요소가 0인 행렬의 복사본을 반환합니다. Upper triangular matrix
Diag	대각선 추출 또는 대각선 행렬 구성
Row	행 벡터를 반환. 지정된 행에 벡터 쓰기
Col	열 벡터를 반환. 지정된 열에 벡터 쓰기
Copy	주어진 행렬/벡터의 복사본을 반환
Concat	1 행렬에 2 개의 하위 매트릭스를 연결합니다. 하나의 벡터에 2개의 벡터를 연결합니다.
Compare	지정된 정밀함으로 두 행렬/벡터의 요소를 비교
CompareByDigits	Significant digits까지 두 행렬/벡터의 요소를 비교합니다.
Flat	두 개가 아닌 하나의 인덱스를 통해 행렬 요소의 주소 지정을 허용합니다.
Clip	행렬/벡터의 요소를 지정된 유효 값 범위로 제한
Reshape	데이터를 변경하지 않고 행렬의 모양 변경
Resize	모양과 크기가 변경된 새 행렬 반환
SwapRows	행렬에서 행을 서로 바꿈
SwapCols	행렬의 열을 바꿈
Split	행렬을 여러 부분 행렬로 분할
Hsplit	행렬을 여러개의 부분 행렬로 수평 분할합니다. 다음과 동일합니다 Split with axis=0
Vsplit	행렬을 세로로 여러개의 부분 행렬로 분할합니다. 다음과 동일합니다 Split with axis=1
ArgSort	행렬 또는 벡터를 간접적으로 정렬합니다.
Sort	행렬 또는 벡터를 제자리에 정렬합니다.

© 2000-2025, MetaQuotes Ltd.

HasNan

행렬/벡터에서 NaN 수를 반환

```
ulong vector::HasNan();

ulong matrix::HasNan();
```

반환 값

NaN 값을 포함하는 행렬/벡터 요소의 수.

주의

NaN 값을 갖는 적절한 요소 쌍을 비교할 때Compare와 CompareByDigits 메서드는 이러한 요소가 같다고 간주하지만 일반적인 부동 소수점 숫자를 비교하는 경우 NaN != NaN입니다.

예:

```
void OnStart(void)
  {
   double x=sqrt(-1);

   Print("single: ",x==x);

   vector<double> v1={x};
   vector<double> v2={x};

   Print("vector: ", v1.Compare(v2,0)==0);
  }

/* Result:

 single: false
 vector: true
*/
```

더 참조할 만한 내용

MathClassify, Compare, CompareByDigits

© 2000-2025, MetaQuotes Ltd.

ReplaceNan

Return the number of NaN values in a matrix/vector.

```
ulong vector::ReplaceNan(
  const double    value      // new value
   );

ulong vectorf::ReplaceNan(
  const float     value      // new value
   );

ulong matrix::ReplaceNan(
  const double    value      // new value
   );

ulong matrixf::ReplaceNan(
  const float     value      // new value
   );
```

Parameters

value

[in] New value to replace NaN elements in the matrix/vector.

Return Value

The number of matrix/vector NaN elements that replaced by new value.

© 2000-2025, MetaQuotes Ltd.

Transpose

매트릭스 전치. 행렬의 축을 뒤집거나 치환합니다; 수정된 행렬을 반환합니다.

```
matrix matrix::Transpose()
```

반환값

전치 행렬.

MQL5의 간단한 행렬 전치 알고리즘:

```
matrix MatrixTranspose(const matrix& matrix_a)
  {
   matrix matrix_c(matrix_a.Cols(),matrix_a.Rows());

   for(ulong i=0; i<matrix_c.Rows(); i++)
      for(ulong j=0; j<matrix_c.Cols(); j++)
         matrix_c[i][j]=matrix_a[j][i];

   return(matrix_c);
  }
```

MQL5 예:

```
  matrix a= {{0, 1, 2}, {3, 4, 5}};
  Print("matrix a \n", a);
  Print("a.Transpose() \n", a.Transpose());

  /*
  matrix a
  [[0,1,2]
   [3,4,5]]
  a.Transpose()
  [[0,3]
   [1,4]
   [2,5]]
  */
```

파이썬 예:

```
import numpy as np

a = np.arange(6).reshape((2,3))
print("a \n",a)
print("np.transpose(a) \n",np.transpose(a))
```

© 2000-2025, MetaQuotes Ltd.

```
a
 [[0 1 2]
 [3 4 5]]
np.transpose(a)
 [[0 3]
 [1 4]
 [2 5]]
```

© 2000-2025, MetaQuotes Ltd.

TransposeConjugate

Transposing a complex matrix with conjugation. Reverse or permute the axes of a matrix by changing a sign of an imaginary part of a complex number, return the modified matrix.

```
matrixc matrixc::TransposeConjugate()
```

Return Value

Transposed complex conjugate matrix.

Note

Conjugate can be applied to real (non-complex) matrix or vector. In this case matrix or vector just copied on return.

Simple algorithm of transposing a complex matrix with conjugation - method explanation

```
//--- Complex matrix transpose function with conjugation
matrixc MatrixTransposeConjugate(const matrixc& matrix_a)
  {
   //--- create a new matrix_c with dimensions inverse to matrix_a
   matrixc matrix_c(matrix_a.Cols(), matrix_a.Rows());

   //--- go through all the rows of the new matrix
   for(ulong i=0; i<matrix_c.Rows(); i++)
     {
      //--- go through all the columns of the new matrix
      for(ulong j=0; j<matrix_c.Cols(); j++)
        {
         //--- transfer the real part of the element by transposing the index
         matrix_c[i][j].real = matrix_a[j][i].real;
         //--- transfer the imaginary part of the element by changing the sign (conjug
         matrix_c[i][j].imag = -matrix_a[j][i].imag;
        }
     }

    //--- return the transposed matrix with conjugation
    return(matrix_c);
  }
```

© 2000-2025, MetaQuotes Ltd.

TriL

k번째 대각선 위의 요소가 0인 행렬의 복사본을 반환 하부 삼각 행렬.

```
matrix matrix::Tril(
  const int      ndiag=0      // 대각선의 인덱스
   );
```

매개 변수

ndiag=0

[in] Diagonal above which to zero elements. ndiag = 0(기본값)은 주 대각선이고 ndiag < 0은 그 아래에 있고 ndiag > 0은 위에 있습니다.

반환값

아래쪽 삼각형이 1로 채워지고 다른 곳은 0으로 채워진 배열입니다.

MQL5 예:

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriL(-1);
   Print("matrix b \n",b);

  /*
  matrix_c
  [[0,0,0]
  [4,0,0]
  [7,8,0]
  [10,11,12]]
  */
```

파이썬 예:

```
import numpy as np

a=np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)

[[ 0  0  0]
 [ 4  0  0]
 [ 7  8  0]
 [10 11 12]]
```

© 2000-2025, MetaQuotes Ltd.

TriU

k 번째 대각선 아래의 요소가 0인 행렬의 복사본을 반환 상부 삼각 행렬.

```
matrix matrix::Triu(
  const int      ndiag=0      // 애각선의 인덱스
   );
```

매개 변수

ndiag=0

[in] Diagonal below which to zero elements. ndiag = 0(기본값)은 주 대각선이고 ndiag < 0은 그 아래에 있고 ndiag > 0은 위에 있습니다.

MQL5 예:

```
   matrix a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   matrix b=a.TriU(-1);
   Print("matrix b \n",b);

  /*
  matrix b
  [[1,2,3]
   [4,5,6]
   [0,8,9]
   [0,0,12]]
  */
```

파이썬 예:

```
import numpy as np

a=np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]], -1)
print(a)

[[ 1  2  3]
 [ 4  5  6]
 [ 0  8  9]
 [ 0  0 12]]
```

© 2000-2025, MetaQuotes Ltd.

Diag

대각선을 추출하거나 대각선 행렬을 구성합니다.

```
vector matrix::Diag(
  const int      ndiag=0      // 대각선의 수
   );

void matrix::Diag(
  const vector   v,           // 대각 벡터
  const int      ndiag=0      // 대각선의 수
   );
```

매개 변수

v

자신의 요소가 해당 대각선에 포함되는 벡터입니다(ndiag=0이 주 대각선임).

ndiag=0

[in] 문제의 대각선. 기본값은 0입니다. 주대각선 위의 대각선에는 ndiag>0을 사용하고 주대각선 아래의 대각선에는 ndiag<0을 사용합니다.

참조

할당되지 않은 행렬(차원이 없음)에 대해 대각선을 설정할 수 있습니다. 이 경우 대각선 벡터의 크기에 해당하는 크기의 제로 행렬이 생성되고 이후 해당 대각선에 벡터 값이 채워집니다. 대각선이 이미 존재하는 행렬로 설정된 경우 행렬 차원은 변경되지 않으며 대각선 벡터 외부의 행렬 요소의 값은 변경되지 않습니다.

예

```
   vector v1={1,2,3};
   matrix m1;
   m1.Diag(v1);
   Print("m1\n",m1);
   matrix m2;
   m2.Diag(v1,-1);
   Print("m2\n",m2);
   matrix m3;
   m3.Diag(v1,1);
   Print("m3\n",m3);
   matrix m4=matrix::Full(4,5,9);
   m4.Diag(v1,1);
   Print("m4\n",m4);

   Print("diag -1 - ",m4.Diag(-1));
   Print("diag 0 - ",m4.Diag());
   Print("diag 1 - ",m4.Diag(1));
```

© 2000-2025, MetaQuotes Ltd.

```
/*

m1
[[1,0,0]
[0,2,0]
[0,0,3]]
m2
[[0,0,0]
[1,0,0]
[0,2,0]
[0,0,3]]
m3
[[0,1,0,0]
[0,0,2,0]
[0,0,0,3]]
m4
[[9,1,9,9,9]
[9,9,2,9,9]
[9,9,9,3,9]
[9,9,9,9,9]]
diag -1 - [9,9,9]
diag 0 - [9,9,9,9]
diag 1 - [1,2,3,9]
*/
```

© 2000-2025, MetaQuotes Ltd.

Row

행 벡터를 반환. 지정된 행에 벡터 쓰기

```
vector matrix::Row(
  const ulong   nrow      // 행 수
   );

void matrix::Row(
  const vector  v,        // 행 벡터
  const ulong   nrow      // 행 수
   );
```

매개 변수

nrow

[in] 행의 수

반환값

Vector.

참조

할당되지 않은 행렬(차원이 없는)에 대해 행이 설정될 수 있습니다. 이 경우 벡터 크기 x 행 번호+1의 크기로 제로 행렬이 생성되고 이후 벡터 요소의 값이 해당 행에 채워집니다. 행이 이미 존재하는 행렬로 설정된 경우 행렬의 차원은 변경되지 않으며 행 벡터 외부의 행렬 요소 값은 변경되지 않습니다.

예

```
   vector v1={1,2,3};
   matrix m1;
   m1.Row(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,7);
   m2.Row(v1,2);
   Print("m2\n",m2);

   Print("row 1 - ",m2.Row(1));
   Print("row 2 - ",m2.Row(2));

  /*
  m1
  [[0,0,0]
  [1,2,3]]
  m2
  [[7,7,7,7,7]
```

© 2000-2025, MetaQuotes Ltd.

```
   [7,7,7,7,7]
   [1,2,3,7,7]
   [7,7,7,7,7]]
   row 1 - [7,7,7,7,7]
   row 2 - [1,2,3,7,7]
   */
```

© 2000-2025, MetaQuotes Ltd.

Col

열 벡터를 반환. 지정된 열에 벡터를 작성.

```
vector matrix::Col(
  const ulong   ncol      // 열 번호
   );

void matrix::Col(
  const vector  v,        // 열 벡터
  const ulong   ncol      // 열 번호
   );
```

매개 변수

ncol

[in] 열의 수

반환값

Vector.

참조

할당되지 않은 행렬(차원이 없는)에 대해 열을 설정할 수 있습니다. 이 경우 벡터 크기 x 열 번호+1의 크기로 제로 행렬이 생성되고 이후 벡터 요소의 값이 해당 열에 채워집니다. 열이 이미 존재하는 행렬로 설정된 경우 행렬 차원은 변경되지 않으며 열 벡터 외부의 행렬 요소 값은 변경되지 않습니다.

예

```
   vector v1={1,2,3};
   matrix m1;
   m1.Col(v1,1);
   Print("m1\n",m1);
   matrix m2=matrix::Full(4,5,8);
   m2.Col(v1,2);
   Print("m2\n",m2);

   Print("col 1 - ",m2.Col(1));
   Print("col 2 - ",m2.Col(2));

  /*
  m1
  [[0,1]
  [0,2]
  [0,3]]
  m2
  [[8,8,1,8,8]
```

© 2000-2025, MetaQuotes Ltd.

```
   [8,8,2,8,8]
   [8,8,3,8,8]
   [8,8,8,8,8]]
   col 1 - [8,8,8,8]
   col 2 - [1,2,3,8]
   */
```

© 2000-2025, MetaQuotes Ltd.

Copy

주어진 행렬/벡터의 복사본을 만듭니다.

```
bool matrix::Copy(
  const matrix&  a     // 복사된 행렬
   );
bool vector::Copy(
  const vector&  v     // 복사된 벡터
   );
```

매개 변수

v

[in] 복사할 행렬 혹은 벡터

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

MQL5 예:

```
  matrix a=matrix::Eye(3, 4);
  matrix b;
  b.Copy(a);
  matrix c=a;
  Print("matrix b \n", b);
  Print("matrix_c \n", c);

  /*
  /*
  matrix b
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  matrix_c
  [[1,0,0,0]
  [0,1,0,0]
  [0,0,1,0]]
  */
  */
```

파이썬 예:

```
import numpy as np

a = np.eye(3,4)
print('a \n',a)
b = a
```

© 2000-2025, MetaQuotes Ltd.

```
print('b \n',b)
c = np.copy(a)
print('c \n',c)

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
b
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
c
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]]
```

© 2000-2025, MetaQuotes Ltd.

Conjugate

Changing the sign of the imaginary part of a complex number, return the modified matrix or vector.

```
matrixc matrixc::Conjugate();

vectorc vectorc::Conjugate();
```

Return Value

Complex conjugated matrix or vector.

Note

Conjugate can be applied to real (non-complex) matrix or vector. In this case matrix or vector just copied on return.

Simple algorithm of conjugation - method explanation

```
//--- Complex vector conjugation
vectorc VectorConjugate(const vectorc& vector_a)
  {
   //--- create a new vector_c
   vectorc vector_c(vector_a.Size());

   //--- go through all values of the new vector
   for(ulong i=0; i<vector_c.Size(); i++)
     {
      //--- transfer the real part of the element
      vector_c[i].real = vector_a[i].real;
      //--- transfer the imaginary part of the element by changing the sign (conjugati
      vector_c[i].imag = -vector_a[i].imag;
     }

    //--- return the conjugated vector
    return(vector_c);
  }
```

© 2000-2025, MetaQuotes Ltd.

Concat

1 행렬에 2 개의 하위 매트릭스를 연결합니다. 하나의 벡터에 2개의 벡터를 연결합니다.

```
vector vector::Concat(
  const vector&  second_part      // 연결할 두 번째 벡터
   );

vector matrix::Concat(
  const matrix&  second_part      // 연결할 두번째 행렬
   );

matrix matrix::Split(
  const matrix&  second_part,     // 연결할 두번째 행렬
  const int      axis             // axis
   );
```

매개 변수

second_part

[in] 연결할 두번째 행렬 혹은 벡터 축을 따라 행렬이 연결되면 축에 대해 행렬의 크기가 일관성이 있어야합니다.

axis

[in] 축. 0 - 수평 축, 1 - 수직 축.

반환 값

벡터 또는 행렬이 축 매개변수 없이 연결된 경우나 수평 또는 수직 축을 따라 행렬이 연결되는 경우 벡터를 반환합니다.

예

```
   vector vector_a={1,2,3,4};
   vector vector_b={5,6,7};
   vector vector_c=vector_a.Concat(vector_b);
   Print("vector_c=",vector_c);

   matrix matrix_a={{1,2,3},{4,5,6}};
   matrix matrix_b={{7,8,9},{10,11,12}};
   vector_c=matrix_a.Concat(matrix_b);
   Print("vector_c=",vector_c);

   matrix matrix_c0=matrix_a.Concat(matrix_b,0);
   Print("matrix_c0=\n",matrix_c0);

   matrix matrix_c1=matrix_a.Concat(matrix_b,1);
   Print("matrix_c1=\n",matrix_c1);

  /*
```

© 2000-2025, MetaQuotes Ltd.

```
vector_c=[1,2,3,4,5,6,7]
vector_c=[1,2,3,4,5,6,7,8,9,10,11,12]
matrix_c0=
[[1,2,3]
 [4,5,6]
 [7,8,9]
 [10,11,12]]
matrix_c1=
[[1,2,3,7,8,9]
 [4,5,6,10,11,12]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Compare

두 행렬/벡터의 요소를 지정된 정밀도로 비교합니다.

```
ulong vector::Compare(
  const vector& vec,          // 비교할 벡터
  const double  epsilon       // 정밀도
   );

ulong matrix::Compare(
  const matrix& mat,          // 비교할 행렬
  const double  epsilon       // 정밀도
   );
```

매개 변수

vector_b

[in] 비교할 벡터

epsilon

[in] 정밀도

반환값

비교 중인 행렬 또는 벡터의 일치하지 않는 요소 수: 행렬이 같으면 0, 그렇지 않으면 0보다 큽니다.

복소수 행렬/벡터의 경우 comparison은 복소수 간의 거리 추정이 포함됩니다. 거리는 sqrt(pow(r1-r2, 2) + pow(i1-i2, 2)로 계산되며 이미 엡실론과 비교할 수 있는 실수입니다.

참조

비교 연산자 == or !=는 정확한 요소별 비교를 실행합니다. 실수의 정확한 비교는 제한적으로 사용되는 것으로 알려져 있으므로 엡실론 비교 방법이 추가되었습니다. 하나의 행렬이 예를 들어 1e-20에서 1e+20까지의 범위에 있는 요소를 포함할 수 있는 경우가 있습니다. 이러한 행렬은 유효 숫자까지 요소별 비교를 사용하여 처리할 수 있습니다.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4}};
   matrix matrix_i=matrix::Identity(3,3);
   matrix matrix_c=matrix_a.Inv();
   matrix matrix_check=matrix_a.MatMul(matrix_c);
   Print("matrix_check\n",matrix_check);

   ulong errors=matrix_check.Compare(matrix::Identity(3,3),1e-15);
   Print("errors=",errors);


  /*
```

© 2000-2025, MetaQuotes Ltd.

```
matrix_check
[[1,0,0]
[4.440892098500626e-16,1,8.881784197001252e-16]
[4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
errors=0

*/
```

© 2000-2025, MetaQuotes Ltd.

CompareByDigits

두 행렬/벡터의 요소를 significant digits precision로 비교합니다.

```
ulong vector::CompareByDigits(
  const vector& vec,          // 비교할 벡터
  const int     digits        // significant digits의 수
   );

ulong matrix::CompareByDigits(
  const matrix& mat,          // 비교할 행렬
  const int     digits        // significant digits의 수
   );
```

매개 변수

vector_b

[in] 비교할 벡터

digits

[in] 비교할 significant digits의 수

epsilon

[in] 비교 정밀도 두 값의 절대값 차이가 지정된 정밀도보다 작은 경우 동일한 것으로 간주됩니다.

반환값

비교 중인 행렬 또는 벡터의 일치하지 않는 요소 수: 행렬이 같으면 0, 그렇지 않으면 0보다 큽니다.

참조

비교 연산자 == or !=는 정확한 요소별 비교를 실행합니다. 실수의 정확한 비교는 제한적으로 사용되는 것으로 알려져 있으므로 엡실론 비교 방법이 추가되었습니다. 하나의 행렬이 예를 들어 1e-20에서 1e+20까지의 범위에 있는 요소를 포함할 수 있는 경우가 있습니다. 이러한 행렬은 significant digits까지 요소별 비교를 사용하여 처리할 수 있습니다.

예

```
   int    size_m=128;
   int    size_k=256;
   matrix matrix_a(size_m,size_k);
//--- 행렬 채우기
   double  value=0.0;
   for(int i=0; i<size_m; i++)
     {
      for(int j=0; j<size_k; j++)
        {
         if(i==j)
            matrix_a[i][j]=1.0+i;
```

© 2000-2025, MetaQuotes Ltd.

```
         else
           {
            value+=1.0;
            matrix_a[i][j]=value/1e+20;
           }
        }
     }
//--- get another matrix
   matrix matrix_c = matrix_a * -1;

   ulong errors_epsilon=matrix_a.Compare(matrix_c,1e-15);
   ulong errors_digits=matrix_a.CompareByDigits(matrix_c,15);

   printf("Compare matrix %d x %d  errors_epsilon=%I64u  errors_digits=%I64u",size_m,s


  /*
  Compare matrix 128 x 256  errors_epsilon=128  errors_digits=32768
  */
```

© 2000-2025, MetaQuotes Ltd.

CompareEqual

Perform an absolute comparison of two matrices by unfolding successive rows into one-dimensional vectors.

```
ulong vector::Compare(
  const vector& vec    // vector to compare
   );

ulong matrix::CompareEqual(
  const matrix& mat     // matrix to compare
   );
```

Parameters

vec

[in] Vector to compare.

mat

[in] Matrix to compare.

Method description

Let us have two matrices: matrix A the method is called for and matrix B, which is passed as a method parameter. The comparison is performed as follows:

1. Matrices are expanded into one-dimensional vectors by successive concatenation of rows.
2. Vectors are compared element by element until the first mismatched element.
3. Depending on the comparison results, one of the values described below is returned.

Return Value

-1 — if the matrix A element is less than the corresponding matrix B element.

0 — if all elements of A and B matrices are identical.

1 — if the matrix A element exceeds the corresponding matrix B element.

Note

NaN value elements are taken into account when comparing.

NaN value elements are considered equal if they are present in both matrices at corresponding positions.

The NaN sign is not taken into account in the comparison.

An element with a NaN value is considered to be less than any other numeric value.

Example

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   matrix matrix_a= {{10, 3, 2}, {1, 8, 12}, {6, 5, 4}};
   matrix matrix_i=matrix::Identity(3, 3);
   matrix matrix_c=matrix_a.Inv();
   matrix matrix_check=matrix_a.MatMul(matrix_c);
   Print("matrix_check\n", matrix_check);

   ulong errors=matrix_check.Compare(matrix::Identity(3, 3), 1e-15);
   Print("errors=", errors);
   /*
   matrix_check
   [[1,0,0]
   [4.440892098500626e-16,1,8.881784197001252e-16]
   [4.440892098500626e-16,2.220446049250313e-16,0.9999999999999996]]
   errors=0
   */

//---  absolute comparison of matrices
   matrix<double> A = matrix_a;  // Matrix A initialization
   matrix<double> B = matrix_c;  // Matrix B initialization
   int result = A.CompareEqual(B);
   switch(result)
     {
      case -1:
         Print("Matrix A is smaller than matrix B");
         break;
      case 0:
         Print("Matrices A and B are identical");
         break;
      case 1:
         Print("Matrix A is greater than matrix B");
         break;
      case -2:
         Print("Error! Matrix A is not initialized");
         break;
      case 2:
         Print("Error! Matrix B is not initialized");
         break;
      case -3:
         Print("Error! The size of matrix A is less than the size of matrix B");
         break;
      case 3:
         Print("Error! The size of matrix A is greater than the size of matrix B");
         break;
      default:
         Print("Error! Unknown error");
```

© 2000-2025, MetaQuotes Ltd.

```
            break;
        }
    }
```

© 2000-2025, MetaQuotes Ltd.

Flat

두 개가 아닌 하나의 인덱스를 통해 행렬 요소의 주소를 지정할 수 있습니다.

```
bool matrix::Flat(
  const ulong   index,      //
  const double  value       // 설정할 값
   );

double matrix::Flat(
  const ulong   index,      //
   );
```

매개 변수

index

[in] Flat 인덱스

value

【n】주어진 인덱스로 설정할 값.

반환값

주어진 인덱스에 의한 값.

참조

행렬 mat(3,3)의 경우 액세스는 다음과 같이 작성할 수 있습니다:

- reading: 'x=mat.Flat(4)', which is equivalent to 'x=mat[1][1]'
- writing: 'mat.Flat(5, 42)', equivalent to 'mat[1][2]=42'

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
   ulong arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));
   matrix_a.Flat(arg_max,0);
   arg_max=matrix_a.ArgMax();
   Print("max_value=",matrix_a.Flat(arg_max));


  /*
  matrix_a
  [[10,3,2]
   [1,8,12]
   [6,5,4]
   [7,11,9]]
```

© 2000-2025, MetaQuotes Ltd.

```
   max_value=12.0
   max_value=11.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Clip

행렬/벡터의 요소를 유효한 값의 지정된 범위로 제한합니다.

```
bool matrix::Clip(
  const double  min_value,     // 최소 값
  const double  max_value      // 최대값
   );
bool vector::Clip(
  const double  min_value,     // 최소 값
  const double  max_value      // 최대값
   );
```

매개 변수

min_value

[in] 최소값

max_value

[in] 최대값

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

행렬(또는 벡터)이 처리됩니다. 복사본이 생성되지 않습니다.

예

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   bool res=matrix_a.Clip(4,8);
   Print("matrix_a\n",matrix_a);

  /*
  matrix_a
  [[4,4,4]
   [4,5,6]
   [7,8,8]
   [8,8,8]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Reshape

데이터를 변경하지 않고 행렬의 모양을 변경합니다.

```
void  Reshape(
  const ulong  rows,     // 새로운 수 혹은 행
  const ulong  cols      // 새로운 수 혹은 열
   );
```

매개 변수

rows

[in] 새로운 수 혹은 행.

cols

[in] 새로운 수 혹은 열.

참조

매트릭스가 처리됩니다. 복사본이 생성되지 않습니다. 모든 크기를 지정할 수 있습니다(즉, rows_new*cols_new!=rows_old*cols_old). 매트릭스 버퍼가 증가하면 추가 값이 정의되지 않습니다.

예

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};

   Print("matrix_a\n",matrix_a);
   matrix_a.Reshape(2,6);
   Print("Reshape(2,6)\n",matrix_a);
   matrix_a.Reshape(3,5);
   Print("Reshape(3,5)\n",matrix_a);
   matrix_a.Reshape(2,4);
   Print("Reshape(2,4)\n",matrix_a);

  /*
  matrix_a
  [[1,2,3]
   [4,5,6]
   [7,8,9]
   [10,11,12]]
  Reshape(2,6)
  [[1,2,3,4,5,6]
   [7,8,9,10,11,12]]
  Reshape(3,5)
  [[1,2,3,4,5]
   [6,7,8,9,10]
   [11,12,0,3,0]]
  Reshape(2,4)
  [[1,2,3,4]
   [5,6,7,8]]
```

© 2000-2025, MetaQuotes Ltd.

```
*/
```

© 2000-2025, MetaQuotes Ltd.

Resize

모양과 크기가 변경된 새 행렬을 반환합니다.

```
bool matrix::Resize(
  const ulong  rows,     //
  const ulong  cols,     // 새로운 수 혹은 열
  const ulong  reserve=0 // 아이템에 어마운트 보관
   );

bool vector::Resize(
  const ulong  size,     // 새로운 사이즈.
  const ulong  reserve=0 // 아이템에 어마운트 보관
   );
```

매개 변수

rows

[in] 새로운 수 혹은 행.

cols

[in] 새로운 수 혹은 열.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

행렬(또는 벡터)이 처리됩니다. 복사본이 생성되지 않습니다. 모든 크기를 지정할 수 있습니다(즉, rows_new*cols_new!=rows_old*cols_old). Reshape와 달리 행렬은 행 단위로 처리됩니다. 열 수를 늘리면 추가 열의 값이 정의되지 않습니다. 행 수를 늘리면 새 행의 요소 값이 정의되지 않습니다. 열 수가 줄어들면 행렬의 각각의 행이 잘립니다.

예

```
   matrix matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);
   matrix_a.Resize(2,6);
   Print("Ressize(2,6)\n",matrix_a);
   matrix_a.Resize(3,5);
   Print("Resize(3,5)\n",matrix_a);
   matrix_a.Resize(2,4);
   Print("Resize(2,4)\n",matrix_a);

   /*
   matrix_a
   [[1,2,3]
    [4,5,6]
```

© 2000-2025, MetaQuotes Ltd.

```
    [7,8,9]
    [10,11,12]]
  Ressize(2,6)
  [[1,2,3,4,5,6]
    [4,5,6,10,11,12]]
  Resize(3,5)
  [[1,2,3,4,5]
    [4,5,6,10,11]
    [11,12,3,8,8]]
  Resize(2,4)
  [[1,2,3,4]
    [4,5,6,10]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Set

지정된 인덱스로 벡터 요소의 값을 정합니다.

```
bool vector::Set(
  ulong   index,     // 요소 인덱스
  double  value      // 값
   );
```

매개 변수

index

[in] 값이 설정되어야 하는 요소의 인덱스.

value

[in] 값.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

Set 메서드는 대괄호를 사용하여 값을 할당하는 것과 동일한 작업을 수행합니다. 즉: *Vector[index]=value*. 이러한 유형의 표기법이 사용되는 언어에서 코드 전송을 단순화하기 위해 메서드가 추가되었습니다. 아래 예는 지정된 인덱스 값으로 벡터를 채우는 두 가지 옵션을 모두 보여줍니다.

예:

```
void OnStart()
  {
//---
   vector v1(10, VectorAssignValues);
   Print("v1 = ", v1);

   vector v2(10, VectorSetValues);
   Print("v2 = ", v2);
  }
 /* Result
  v1 = [1,2,4,8,16,32,64,128,256,512]
  v2 = [1,2,4,8,16,32,64,128,256,512]
  */
//+------------------------------------------------------------------+
//| 할당 작업을 통해 벡터를 숫자의 거듭제곱으로 채웁니다.  |
//+------------------------------------------------------------------+
void VectorAssignValues(vector& v, double initial=1)
  {
   double value=initial;
   for(ulong k=0; k<v.Size(); k++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       v[k]=value;
       value*=2;
      }
   }
//+------------------------------------------------------------------------+
//| Set 메서드를 사용하여 숫자의 거듭제곱으로 벡터를 채웁니다.                |
//+------------------------------------------------------------------------+
void VectorSetValues(vector& v, double initial=1)
   {
    double value=initial;
    for(ulong k=0; k<v.Size(); k++)
      {
       v.Set(k, value);
       value*=2;
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

SwapRows

행렬의 행을 바꿉니다.

```
bool matrix::SwapRows(
  const ulong  row1,     // 첫번째 행의 인덱스
  const ulong  row2      // 두번째 행의 인덱스
   );
```

매개 변수

row1

[in] 첫번째 행의 인덱스.

row2

[in] 두번째 행의 인덱스.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

예

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapRows(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapRows(0,3);
   matrix matrix_c1=matrix_p.MatMul(matrix_a);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  matrix_c1
  [[13,14,15,16]
   [5,6,7,8]
   [9,10,11,12]
   [1,2,3,4]]
  */
```

© 2000-2025, MetaQuotes Ltd.

SwapCols

행렬의 열을 바꿉니다.

```
bool matrix::SwapCols(
  const ulong  row1,     // 첫번째 열의 인덱스
  const ulong  row2      // 두번째 열의 인덱스
   );
```

매개 변수

col1

[in] 첫번째 열의 인덱스.

col2

[in] 두번째 열의 인덱스

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

예

```
   matrix matrix_a={{1,2,3,4},
                    {5,6,7,8},
                    {9,10,11,12},
                    {13,14,15,16}};
   matrix matrix_i=matrix::Identity(4,4);
   matrix matrix_a1=matrix_a;
   matrix_a1.SwapCols(0,3);
   Print("matrix_a1\n",matrix_a1);

   matrix matrix_p=matrix_i;
   matrix_p.SwapCols(0,3);
   matrix matrix_c1=matrix_a.MatMul(matrix_p);
   Print("matrix_c1\n",matrix_c1);

  /*
  matrix_a1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  matrix_c1
  [[4,2,3,1]
   [8,6,7,5]
   [12,10,11,9]
   [16,14,15,13]]
  */
```

© 2000-2025, MetaQuotes Ltd.

Split

행렬을 여러 부분 행렬로 분할합니다.

```
bool matrix::Split(
  const ulong  parts,      // 부분 행렬의 수
  const int    axis,       // axis
  matrix&      splitted[]  // 결과 부분 행렬의 배열
   );

void matrix::Split(
  const ulong& parts[],    부분 행렬의 크기
  const int    axis,       // axis
  matrix&      splitted[]  // 결과 부분 행렬의 배열
   );
```

매개 변수

parts

【n】행렬을 나눌 부분 행렬의 수입니다.

axis

[in] Axis. 0 - 수평 축, 1 - 수직 축.

splitted

【out】결과 부분 행렬의 배열입니다.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

부분 행렬의 수가 지정되면 동일한 크기의 부분 행렬을 얻습니다. 즉, 행렬 크기(0 - 행 수, 1 - 열 수)는 나머지 없이 '부분'으로 나눌 수 있어야 합니다. 다양한 크기의 부분 행렬은 부분 행렬 크기의 배열을 사용하여 얻을 수 있습니다. 크기 배열의 요소는 전체 행렬이 분할될 때까지 사용됩니다. 크기 배열이 종료되고 행렬이 아직 완전히 분할되지 않은 경우 분할되지 않은 나머지가 마지막 부분 행렬이 됩니다.

예

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,2};
```

© 2000-2025, MetaQuotes Ltd.

```
 bool res=matrix_a.Split(2,0,splitted);
 Print(res,"  ",GetLastError());
 ResetLastError();
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(2,1,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

 res=matrix_a.Split(parts,0,splitted);
 Print(res,"  ",GetLastError());
 for(uint i=0; i<splitted.Size(); i++)
    Print("splitted ",i,"\n",splitted[i]);

/*
false  4003
true  0
splitted 0
[[1,2,3]
 [7,8,9]
 [13,14,15]
 [19,20,21]
 [25,26,27]]
splitted 1
[[4,5,6]
 [10,11,12]
 [16,17,18]
 [22,23,24]
 [28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
 [7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
 [19,20,21,22,23,24]]
splitted 2
[[25,26,27,28,29,30]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Hsplit

행렬을 여러 부분 행렬로 수평 분할 axis=0으로 Split과 동일

```
bool matrix::Hsplit(
  const ulong  parts,      // 부분 행렬의 수
  matrix&      splitted[]  // 결과 부분 행렬의 배열
   );

void matrix::Hsplit(
  const ulong& parts[],    부분 행렬의 크기
  matrix&      splitted[]  // 결과 부분 행렬의 배열
   );
```

매개 변수

parts

【n】행렬을 나눌 부분 행렬의 수입니다.

splitted

【but】결과 부분 행렬의 배열입니다.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

부분 행렬의 수가 지정되면 동일한 크기의 부분 행렬을 얻습니다. 행의 개수는 나머지 없이 '부분'으로 나눌 수 있어야 합니다. 다양한 크기의 부분 행렬은 부분 행렬 크기의 배열을 사용하여 얻을 수 있습니다. 크기 배열의 요소는 전체 행렬이 분할될 때까지 사용됩니다. 크기 배열이 종료되고 행렬이 아직 완전히 분할되지 않은 경우 분할되지 않은 나머지가 마지막 부분 행렬이 됩니다.

예

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18},
                    {19,20,21,22,23,24},
                    {25,26,27,28,29,30}};
   matrix splitted[];
   ulong  parts[]={2,4};

   bool res=matrix_a.Hsplit(2,splitted);
   Print(res,"  ",GetLastError());
   ResetLastError();
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);
```

© 2000-2025, MetaQuotes Ltd.

```
   res=matrix_a.Hsplit(5,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   res=matrix_a.Hsplit(parts,splitted);
   Print(res,"  ",GetLastError());
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);


/*
false  4003
true  0
splitted 0
[[1,2,3,4,5,6]]
splitted 1
[[7,8,9,10,11,12]]
splitted 2
[[13,14,15,16,17,18]]
splitted 3
[[19,20,21,22,23,24]]
splitted 4
[[25,26,27,28,29,30]]
true  0
splitted 0
[[1,2,3,4,5,6]
[7,8,9,10,11,12]]
splitted 1
[[13,14,15,16,17,18]
[19,20,21,22,23,24]
[25,26,27,28,29,30]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Vsplit

행렬을 세로로 여러 부분 행렬로 분할 axis=1로 Split과 동일

```
bool matrix::Vsplit(
  const ulong  parts,      // 부분 행렬의 수
  matrix&      splitted[]  // 결과 부분 행렬의 배열
   );

void matrix::Vsplit(
  const ulong& parts[],    부분 행렬의 크기
  matrix&      splitted[]  // 결과 부분 행렬의 배열
   );
```

매개 변수

parts

【n】행렬을 나눌 부분 행렬의 수입니다.

splitted

【but】결과 부분 행렬의 배열입니다.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

부분 행렬의 수가 지정되면 동일한 크기의 부분 행렬을 얻습니다. 열의 개수는 나머지 없이 '부분'으로 나눌 수 있어야 함을 의미합니다. 다양한 크기의 부분 행렬은 부분 행렬 크기의 배열을 사용하여 얻을 수 있습니다. 크기 배열의 요소는 전체 행렬이 분할될 때까지 사용됩니다. 크기 배열이 종료되고 행렬이 아직 완전히 분할되지 않은 경우 분할되지 않은 나머지가 마지막 부분 행렬이 됩니다.

예

```
   matrix matrix_a={{ 1, 2, 3, 4, 5, 6},
                    { 7, 8, 9,10,11,12},
                    {13,14,15,16,17,18}};
   matrix splitted[];
   ulong  parts[]={2,3};

   matrix_a.Vsplit(2,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   matrix_a.Vsplit(3,splitted);
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);

   matrix_a.Vsplit(parts,splitted);
```

© 2000-2025, MetaQuotes Ltd.

```
   for(uint i=0; i<splitted.Size(); i++)
      Print("splitted ",i,"\n",splitted[i]);


  /*
     splitted 0
     [[1,2,3]
      [7,8,9]
      [13,14,15]]
     splitted 1
     [[4,5,6]
      [10,11,12]
      [16,17,18]]

     splitted 0
     [[1,2]
      [7,8]
      [13,14]]
     splitted 1
     [[3,4]
      [9,10]
      [15,16]]
     splitted 2
     [[5,6]
      [11,12]
      [17,18]]

     splitted 0
     [[1,2]
      [7,8]
      [13,14]]
     splitted 1
     [[3,4,5]
      [9,10,11]
      [15,16,17]]
     splitted 2
     [[6]
      [12]
      [18]]

  */
```

© 2000-2025, MetaQuotes Ltd.

ArgSort

행렬 또는 벡터의 간접 정렬입니다.

```
vector vector::Sort(
  func_name  compare_func=NULL,  // 비교 함수
  T context                        // 사용자 지정 정렬 함수의 매개변수
   );

matrix matrix::Sort(
  func_name  compare_func=NULL   // 비교함수
  T context // 사용자 지정 정렬 함수의 매개변수
   );

matrix matrix::Sort(
  const int  axis,               // 정렬을 위한 축
  func_name  compare_func=NULL   // 비교함수
  T context // 사용자 지정 정렬 함수의 매개변수
   );
```

매개 변수

axis

【n】정렬할 축: 0은 수평, 1은 수직입니다.

func_name

[in] Comparator. 다음값 중 하나를 지정할 수 있습니다. ENUM_SORT_MODE 열거 또는 자신의 비교 함수. 함수를 지정하지 않으면 오름차순 정렬이 사용됩니다.

사용자 지정 비교 함수는 두 가지 유형이 될 수 있습니다.

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

여기서 T는 행렬 또는 벡터의 유형이고 TContex는 Sort 메서드에 추가 매개 변수로 전달되는 'context' 변수의 유형입니다.

context

【n】사용자 지정 정렬 함수에 전달할 수 있는 추가적인 선택적 매개 변수입니다.

반환값

정렬된 요소의 인덱스가 있는 벡터 또는 행렬입니다. 예를 들어 결과 [4,2,0,1,3]은 0 위치에 인덱스 4가 있는 요소, 첫 번째 위치에 인덱스 2가 있는 요소 등이 있어야 함을 나타냅니다.

© 2000-2025, MetaQuotes Ltd.

Sort

행렬 또는 벡터를 제자리에 정렬합니다.

```
void vector::Sort(
  func_name  compare_func=NULL,  // 비교 함수
T context // 사용자 지정 정렬 함수의 매개변수
   );

void matrix::Sort(
  func_name  compare_func=NULL   // 비교함수
T context // 사용자 지정 정렬 함수의 매개변수
   );

void matrix::Sort(
  const int  axis,               // 정렬을 위한 축
  func_name  compare_func=NULL   // 비교함수
T context // 사용자 지정 정렬 함수의 매개변수
   );
```

매개 변수

axis

【n】정렬할 축: 0은 수평, 1은 수직입니다.

func_name

[in] Comparator. 다음값 중 하나를 지정할 수 있습니다. ENUM_SORT_MODE 열거 또는 자신의 비교 함수. 함수를 지정하지 않으면 오름차순 정렬이 사용됩니다.

사용자 지정 비교 함수는 두 가지 유형이 될 수 있습니다.

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

여기서 T는 행렬 또는 벡터의 유형이고 TContex는 Sort 메서드에 추가 매개 변수로 전달되는 'context' 변수의 유형입니다.

context

【n】사용자 지정 정렬 함수에 전달할 수 있는 추가적인 선택적 매개 변수입니다.

반환값

None. 정렬은 제자리에서 수행됩니다. 즉, Sort 메서드가 호출되는 행렬/벡터의 데이터에 적용됩니다.

예

```
//+------------------------------------------------------------------+
//| Sort function                                                    |
//+------------------------------------------------------------------+
int MyDoubleComparator(double x1,double x2,int sort_mode=0)
  {
   int res=x1<x2 ? -1 : (x1>x2 ? 1 : 0);
```

© 2000-2025, MetaQuotes Ltd.

```
   return(sort_mode==0 ? res : -res);
  }
//+------------------------------------------------------------------+
//| Script start function                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- 벡터 채우기
   vector v(100);
   //--- 오름차순 정렬
   v.Sort(MyDoubleComparator);   // 기본값이 '0'인 추가 매개변수가 여기에 사용됩니다.
   Print(v);
   // 내림차순 정렬
   v.Sort(MyDoubleComparator,1); // 여기서 추가 매개변수 '1'은 사용자가 명시적으로 지정합니다
   Print(v);
  }
```

© 2000-2025, MetaQuotes Ltd.

행렬과 벡터를 사용한 수학 연산

덧셈, 뺄셈, 곱셈 및 나눗셈을 포함한 수학 연산을 행렬 및 벡터 요소별로 수행할 수 있습니다.

수학 함수는 원래 스칼라 값에 관련 작업을 수행하도록 설계되었습니다. 대부분의 기능을 행렬과 벡터에 적용할 수 있습니다. 다음을 포함합니다. MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, and MathTanh. 이러한 연산은 행렬과 벡터의 요소별 처리를 의미합니다. Example

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

MathMod 그리고MathPow 의 경우 두 번째 요소는 적절한 크기의 스칼라 또는 행렬/벡터일 수 있습니다.

다음 예제는 수학 함수를 벡터에 적용하여 표준 편차를 계산하는 방법을 보여줍니다.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- 초기화 함수를 사용하여 벡터를 채웁니다.
  vector r(10, ArrayRandom); // 0에서 1까지의 난수 배열
//--- mean값 계산
  double avr=r.Mean();       // mean 값 배열
  vector d=r-avr;            // mean에서 편차의 배열을 계산
  Print("avr(r)=", avr);
  Print("r=", r);
  Print("d=", d);
  vector s2=MathPow(d, 2);   // 제곱 편차의 배열
  double sum=s2.Sum();       // 제곱 편차의 합
//--- 두 가지 다른 방법으로 표준 편차를 계산합니다.
  double std=MathSqrt(sum/r.Size());
  Print(" std(r)=", std);
  Print("r.Std()=", r.Std());
 }
/*
  avr(r)=0.5300302133243813
```

© 2000-2025, MetaQuotes Ltd.

```
   r=[0.8346201971495713,0.8031556138798182,0.6696676534318063,0.05386516922513505,0.54
   d=[0.30458998382519,0.2731254005554369,0.1396374401074251,-0.4761650440992462,0.0190
    std(r)=0.2838269732183663
   r.Std()=0.2838269732183663
*/
//+------------------------------------------------------------------+
//| 벡터를 랜덤 넘버로 채우기                                        |
//+------------------------------------------------------------------+
void ArrayRandom(vector& v)
 {
  for(ulong i=0; i<v.Size(); i++)
    v[i]=double(MathRand())/32767.;
 }
```

© 2000-2025, MetaQuotes Ltd.

수학 연산

덧셈, 뺄셈, 곱셈 및 나눗셈을 포함한 수학 연산을 행렬 및 벡터 요소별로 수행할 수 있습니다.

두 행렬 또는 두 벡터 모두 유형이 동일하고 차원이 동일해야 합니다. 행렬의 각 요소는 두 번째 행렬의 해당 요소에서 작동합니다.

적절한 유형(double, float 또는 complex)의 스칼라를 두 번째 term(multiplier, subtrahend 또는 divisor)로 사용할 수도 있습니다. 이 경우 행렬 또는 벡터의 각 멤버는 지정된 스칼라에서 작동합니다.

```
   matrix matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
   matrix matrix_b={{1,2,3},{4,5,6}};

   matrix matrix_c1=matrix_a+matrix_b;
   matrix matrix_c2=matrix_b-matrix_a;
   matrix matrix_c3=matrix_a*matrix_b;    // Hadamard 곱, 행렬곱과 혼동하지 마세요! 이를 위한
   matrix matrix_c4=matrix_b/matrix_a;

   matrix_c1=matrix_a+1;
   matrix_c2=matrix_b-double_value;
   matrix_c3=matrix_a*M_PI;
   matrix_c4=matrix_b/0.1;

//--- 작업이 가능합니다.
   matrix_a+=matrix_b;
   matrix_a/=2;
```

벡터에 대해 동일한 작업을 사용할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

수학 함수

다음 수학 함수를 행렬 및 벡터에 적용할 수 있습니다: MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, MathTanh. 이러한 연산은 행렬과 벡터의 요소별 처리를 의미합니다.

MathMod 및 MathPow의 경우 두 번째 요소는 적절한 크기의 스칼라 또는 행렬/벡터일 수 있습니다.

```
   matrix<T> mat1(128,128);
   matrix<T> mat3(mat1.Rows(),mat1.Cols());
   ulong     n,size=mat1.Rows()*mat1.Cols();
...
   mat2=MathPow(mat1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),(T)1.9);
      if(res!=mat2.Flat(n))
         errors++;
     }
   mat2=MathPow(mat1,mat3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(mat1.Flat(n),mat3.Flat(n));
      if(res!=mat2.Flat(n))
         errors++;
     }
   ...
   vector<T> vec1(16384);
   vector<T> vec3(vec1.Size());
   ulong     n,size=vec1.Size();
   ...
   vec2=MathPow(vec1,(T)1.9);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],(T)1.9);
      if(res!=vec2[n])
         errors++;
     }
   vec2=MathPow(vec1,vec3);
   for(n=0; n<size; n++)
     {
      T res=MathPow(vec1[n],vec3[n]);
      if(res!=vec2[n])
         errors++;
     }
```

© 2000-2025, MetaQuotes Ltd.

행렬 및 벡터 곱

행렬 및 벡터 곱 계산에는 다음이 포함됩니다:

- 행렬 곱셈
- 벡터 곱셈
- 공분산 행렬 계산
- 두 벡터의 상호 상관 계산
- 두 벡터의 컨벌루션 계산
- 상관 계수 계산

함수	액션
MatMul	두 행렬의 행렬 곱
GeMM	일반 행렬 곱셈(GeMM)
Power	정사각 행렬을 정수 거듭제곱으로 올립니다.
Dot	두 벡터의 내적
Kron	두 행렬, 행렬과 벡터, 벡터와 행렬 또는 두 벡터의 Kronecker 곱을 반환
Inner	두 행렬의 내적
Outer	두 행렬 또는 두 벡터의 외적 계산
CorrCoef	Pearson 상관 계수 계산(선형 상관 계수)
Cov	공분산 행렬 계산
Correlate	두 벡터의 상호 상관을 계산
Convolve	두 벡터의 불연속 선형 컨벌루션을 반환

© 2000-2025, MetaQuotes Ltd.

MatMul

행렬과 벡터의 곱셈을 가능하게 하는 MatMul 메서드에는 몇 가지 오버로드가 있습니다.

Multiplying a matrix by a matrix: matrix[M][K] * matrix[K][N] = matrix[M][N]

```
matrix  matrix::MatMul(
  const matrix&  b      // second matrix
   );
```

Multiplying a vector by a matrix: horizontal vector[K] * matrix[K][N] = horizontal vector[N]

```
vector  vector::MatMul(
  const matrix&  b      // matrix
   );
```

Multiplying a matrix by a vector: matrix[M][K] * vertical vector[K] = vertical vector[M]

```
vector  matrix::MatMul(
  const vector&  b      // vector
   );
```

Scalar vector multiplication: horizontal vector * vertical vector = dot value

```
scalar  vector::MatMul(
  const vector&  b      // second vector
   );
```

매개 변수

b

[in] 행렬 혹은 벡터.

반환값

사용된 메서드에 따라 행렬, 벡터 또는 스칼라.

참조

행렬은 곱셈과 호환되어야 합니다. 즉, 첫 번째 행렬의 열 수가 두 번째 행렬의 행 수와 같아야 합니다. 행렬 곱셈은 비가환적입니다. 일반적인 경우 첫 번째 행렬에 두 번째 행렬을 곱한 결과는 두 번째 행렬에 첫 번째 행렬을 곱한 결과와 같지 않습니다.

행렬 곱은 첫 번째 행렬의 행 벡터와 두 번째 행렬의 열 벡터의 스칼라 곱의 가능한 모든 조합으로 구성됩니다.

스칼라 곱셈에서 벡터는 길이가 같아야 합니다.

벡터와 행렬을 곱할 때 벡터의 길이는 행렬의 열 수와 정확히 일치해야 합니다.

MQL5의 간단한 행렬 곱셈 알고리즘:

© 2000-2025, MetaQuotes Ltd.

```
matrix MatrixProduct(const matrix& matrix_a, const matrix& matrix_b)
  {
   matrix matrix_c;

   if(matrix_a.Cols()!=matrix_b.Rows())
      return(matrix_c);

   ulong M=matrix_a.Rows();
   ulong K=matrix_a.Cols();
   ulong N=matrix_b.Cols();
   matrix_c=matrix::Zeros(M,N);

   for(ulong m=0; m<M; m++)
      for(ulong k=0; k<K; k++)
         for(ulong n=0; n<N; n++)
            matrix_c[m][n]+=matrix_a[m][k]*matrix_b[k][n];

   return(matrix_c);
  }
```

행렬 곱셈 예제

```
   matrix a={{1, 0, 0},
             {0, 1, 0}};
   matrix b={{4, 1},
             {2, 2},
             {1, 3}};
   matrix c1=a.MatMul(b);
   matrix c2=b.MatMul(a);
   Print("c1 = \n", c1);
   Print("c2 = \n", c2);
/*
   c1 =
   [[4,1]
    [2,2]]
   c2 =
   [[4,1,0]
    [2,2,0]
    [1,3,0]]
*/
```

수평 벡터에 행렬을 곱하는 예

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 3x5 행렬 생성
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   vector v3 = {1, 2, 3};
   Print("Product of horizontal vector v and matrix m[3,5]");
   Print("On the left, vector v3 = ", v3);
   Print("On the right, matrix m35 = \n", m35);
   Print("v3.MatMul(m35) = horizontal vector v[5] \n", v3.MatMul(m35));

  /* Result
    Product of horizontal vector v3 and matrix m[3,5]
    On the left, vector v3 = [1,2,3]
    On the right, matrix m35 =
    [[0,1,2,3,4]
     [5,6,7,8,9]
     [10,11,12,13,14]]
    v3.MatMul(m35) = horizontal vector v[5]
    [40,46,52,58,64]
   */
  }
//+------------------------------------------------------------------+
//|  Fill the matrix with increasing values                          |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

행렬에 수직 벡터를 곱하는 방법의 예

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 3x5 행렬 생성
   matrix m35;
   m35.Init(3, 5, Arange);
//---
   Print("Product of matrix m[3,5] and vertical vector v[5]");
   vector v5 = {1,2,3,4,5};
   Print("On the left, m35 = \n",m35);
   Print("On the right v5 = ",v5);
   Print("m35.MatMul(v5) = vertical vector v[3] \n",m35.MatMul(v5));

  /* Result
   Product of matrix m[3,5] and vertical vector v[5]
   On the left, m35 =
   [[0,1,2,3,4]
    [5,6,7,8,9]
    [10,11,12,13,14]]
   On the right, v5 = [1,2,3,4,5]
   m35.MatMul(v5) = vertical vector v[3]
   [40,115,190]
   */
  }
//+------------------------------------------------------------------+
//|  Fill the matrix with increasing values                          |
//+------------------------------------------------------------------+
void Arange(matrix & m, double start = 0, double step = 1)
  {
//---
   ulong cols = m.Cols();
   ulong rows = m.Rows();
   double value = start;
   for(ulong r = 0; r < rows; r++)
     {
      for(ulong c = 0; c < cols; c++)
        {
         m[r][c] = value;
         value += step;
        }
     }
//---
  }
```

벡터의 스칼라(dot) product의 예

```
void OnStart()
  {
//--- 수평 벡터와 수직 벡터의 스칼라 곱
   vector a= {1, 2, 3};  // horizontal vector
```

© 2000-2025, MetaQuotes Ltd.

```
   vector b= {4, 5, 6};  // vertical vector
   Print("a = ", a);
   Print("b = ", b);
   Print("1) a.MatMul(b) = ", a.MatMul(b));
//--- Dot 메서드가 동일한 결과를 생성하는지 확인합니다.
   Print("2) a.Dot(b) = ", a.Dot(b));

  /* Result
   a = [1,2,3]
   b = [4,5,6]
   1) a.MatMul(b) = 32.0
   2) a.Dot(b) = 32.0
    */
  }
```

또다른 참조

Dot, GeMM

© 2000-2025, MetaQuotes Ltd.

GeMM

GeMM(General Matrix Multiply) 메서드는 두 행렬의 일반적인 곱셈을 구현합니다. 연산은 C ← α AB + β C로 정의되며 여기서 A 및 B 행렬은 선택적으로 전치될 수 있습니다. 행렬 AB (MatMul), *알파* 스칼라는 1과 같다고 가정하고 *베타*는 0과 같습니다.

효율성 측면에서 GeMM과 MatMul의 주요 차이점은 MatMul은 항상 새로운 행렬/벡터 객체를 생성하는 반면 GeMM은 기존 행렬 객체와 함께 작동하며 행렬을 다시 생성하지 않는다는 것입니다. 따라서 GeMM을 사용하고 해당 행렬에 대해 메모리를 사전 할당하면 동일한 행렬 크기로 작업하는 동안 메모리를 재할당하지 않습니다. 이는 예를 들어 전략 테스터에서 최적화를 실행하거나 신경망을 교육할 때 대량 컴퓨팅을 위한 GeMM의 중요한 이점이 될 수 있습니다.

MatMul과 유사하게 GeMM에도 4개의 과부하가 있습니다. 그러나 수직 벡터와 a 및 수평 벡터의 곱셈을 가능하게 하기 위해 네 번째 오버로드의 의미 체계가 수정되었습니다.

기존 행렬/벡터 개체에서는 메모리를 미리 할당할 필요가 없습니다. 메모리는 첫 번째 GeMM 호출에서 할당되고 0으로 채워집니다.

Multiplying a matrix by a matrix: `matrix C[M][N] = α * ( matrix A[M][K] * matrix B[K][N]) + β * matrix C[M][N]`

```
bool  matrix::GeMM(
  const matrix &A,     // 첫번째 행렬
  const matrix &B,     // 두번째 행렬
  double alpha,        // 행렬곱에 대한 알파 승수
  double beta,         //  행렬 C에 대한 베타 승수
  uint   flags         // 행렬 A, B 및 C의 전치 여부를 결정하는 ENUM_GEMM 열거형 값
   );
```

Multiplying a vector by a matrix: `vector C[N] = α * ( vector A[K] * matrix B[K][N]) + β * vector C[N]`

```
bool  matrix::GeMM(
  const vector &A,     // 수평 벡터
  const matrix &B,     // 행렬
  double alpha,        // 행렬곱에 대한 알파 승수
  double beta,         // 벡터 C에 대한 베타 승수
  uint   flags         //  A, B, C의 전치 여부를 결정하는 ENUM_GEMM 열거형 값
   );
```

Multiplying a matrix by a vector: `vector C[M] = α * ( matrix A[M][K] * vector B[K] * ) + β * vector C[M]`

```
bool  matrix::GeMM(
  const matrix &A,     // 행렬
  const vector &B,     // 수직 벡터
  double alpha,        // 행렬곱에 대한 알파 승수
  double beta,         // 벡터 C에 대한 베타 승수
  uint   flags         //  A, B, C의 전치 여부를 결정하는 ENUM_GEMM 열거형 값
   );
```

© 2000-2025, MetaQuotes Ltd.

Multiplying a vector by a vector: `matrix C[M][N] = α * ( vector A[M] * vector B[N] * ) + β * matrix C[M][N]`. 이 오버로드는 스칼라를 반환하는 MatMul과 달리 행렬을 반환합니다.

```
bool  matrix::GeMM(
  const vector &A,     // 첫번째 벡터
  const vector &B,     // 두번째 벡터
  double alpha,        // 행렬곱에 대한 알파 승수
  double beta,         // matrix C에 대한 벡터
  uint   flags         //  A, B, C의 전치 여부를 결정하는 ENUM_GEMM 열거형 값
   );
```

매개 변수

A

[in] 헹렬 혹은 벡터.

B

[in] 헹렬 혹은 벡터.

alpha

[in] 행렬 곱셉 AB에 대한 알파 승수.

beta

[in] 행렬 C에 대한 베타 승수.

flags

【n】행렬 A, B 및 C가 전치되는지 여부를 결정하는 ENUM_GEMM 열거형 값입니다.

반환값

성공하면 true를 그렇지 않으면 false를 반환.

ENUM_GEMM

GeMM 메서드에 대한 플래그의 열거.

ID	설명
TRANSP_A	전치 행렬 A 사용
TRANSP_B	전치 행렬 B 사용
TRANSP_C	전치 행렬 C 사용

참조

float, double 및 complex 유형의 행렬 및 벡터를 매개변수 A 및 B로 사용할 수 있습니다. GeMM 메서드의 템플릿 변형은 다음과 같습니다.

```
bool matrix<T>::GeMM(const matrix<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool matrix<T>::GeMM(const vector<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)
```

© 2000-2025, MetaQuotes Ltd.

```
bool vector<T>::GeMM(const vector<T> &A,const matrix<T> &B,T alpha,T beta,ulong flags)
bool vector<T>::GeMM(const matrix<T> &A,const vector<T> &B,T alpha,T beta,ulong flags)
```

기본적으로 일반적인 행렬 곱셈 함수는 다음과 같이 설명됩니다:

```
C[m,n] = α*Sum(A[m,k]*B[k,n]) + β*C[m,n]
```

크기는 다음과 같습니다. 행렬 A는 M x K, 행렬 B는 K x N, 행렬 C는 M x N입니다.

따라서 행렬은 곱셈과 호환되어야 합니다. 즉 첫 번째 행렬의 열 수가 두 번째 행렬의 행 수와 같아야 합니다. 행렬 곱셈은 비가환적입니다. 일반적인 경우 첫 번째 행렬에 두 번째 행렬을 곱한 결과는 두 번째 행렬에 첫 번째 행렬을 곱한 결과와 같지 않습니다.

예:

```
void OnStart()
  {
   vector vector_a= {1, 2, 3, 4, 5};
   vector vector_b= {4, 3, 2, 1};
   matrix matrix_c;
//--- 두 벡터에 대한 GeMM 계산
   matrix_c.GeMM(vector_a, vector_b, 1, 0);
   Print("matrix_c:\n ", matrix_c, "\n");
   /*
   matrix_c:
    [[4,3,2,1]
    [8,6,4,2]
    [12,9,6,3]
    [16,12,8,4]
    [20,15,10,5]]
   */
//--- 행렬을 벡터로 생성
   matrix matrix_a(5, 1);
   matrix matrix_b(1, 4);
   matrix_a.Col(vector_a, 0);
   matrix_b.Row(vector_b, 0);
   Print("matrix_a:\n ", matrix_a);
   Print("matrix_b:\n ", matrix_b);
   /*
   matrix_a:
   [[1]
   [2]
   [3]
   [4]
   [5]]
   matrix_b:
```

© 2000-2025, MetaQuotes Ltd.

```
    [[4,3,2,1]]
    */
//-- 두 행렬에 대해 GeMM을 계산하고 동일한 결과를 얻습니다.
    matrix_c.GeMM(matrix_a, matrix_b, 1, 0);
    Print("matrix_c:\n ", matrix_c);
    /*
    matrix_c:
     [[4,3,2,1]
     [8,6,4,2]
     [12,9,6,3]
     [16,12,8,4]
     [20,15,10,5]]
    */
  }
```

또다른 참조

MatMul

© 2000-2025, MetaQuotes Ltd.

Power

정사각 행렬을 정수 거듭제곱으로 올립니다.

```
matrix matrix::Power(
  const int  power      // power
   );
```

매개 변수

power

【n】지수는 정수, 양수, 음수 또는 0일 수 있습니다.

반환값

행렬.

참조

결과 행렬은 원래 행렬과 크기가 같습니다. 0의 거듭제곱으로 증가하면 항등 행렬이 반환됩니다. 양의 거듭제곱 n은 원래 행렬이 자체적으로 n번 곱해짐을 의미합니다. 음의 거듭제곱 -n은 원래 행렬이 먼저 반전된 다음 반전된 행렬에 n번 곱해짐을 의미합니다.

MQL5에서 행렬을 거듭제곱하는 간단한 알고리즘:

```
bool MatrixPower(matrix& c, const matrix& a, const int power)
  {
//--- 행렬은 정사각형이어야 합니다.
   if(a.Rows()!=a.Cols())
      return(false);
//--- 결과 행렬의 크기는 정확히 동일합니다.
   ulong  rows=a.Rows();
   ulong  cols=a.Cols();
   matrix result(rows,cols);
//--- 0이 되면 항등 행렬을 반환합니다.
   if(power==0)
      result.Identity();
   else
     {
//--- 음의 차수의 경우 먼저 행렬을 반전시킵니다.
      if(power<0)
        {
         matrix inverted=a.Inv();
         result=inverted;
         for(int i=-1; i>power; i--)
            result=result.MatMul(inverted);
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          result=a;
          for(int i=1; i<power; i++)
             result=result.MatMul(a);
         }
      }
//---
   c=result;
   return(true);
  }
```

MQL5 예:

```
  matrix i= {{0, 1}, {-1, 0}};
  Print("i:\n", i);

  Print("i.Power(3):\n", i.Power(3));

  Print("i.Power(0):\n", i.Power(0));

  Print("i.Power(-3):\n", i.Power(-3));

  /*
  i:
  [[0,1]
   [-1,0]]

  i.Power(3):
  [[0,-1]
   [1,0]]

  i.Power(0):
  [[1,0]
   [0,1]]

  i.Power(-3):
  [[0, -1]
   [1,0]]
  */
```

파이썬 예:

```
import numpy as np
from numpy.linalg import matrix_power

# matrix equiv. of the imaginary unit
i = np.array([[0, 1], [-1, 0]])
```

© 2000-2025, MetaQuotes Ltd.

```
print("i:\n",i)

# should = -i
print("matrix_power(i, 3) :\n",matrix_power(i, 3) )

print("matrix_power(i, 0):\n",matrix_power(i, 0))

# should = 1/(-i) = i, but w/ f.p. elements
print("matrix_power(i, -3):\n",matrix_power(i, -3))

i:
 [[ 0  1]
 [-1  0]]

matrix_power(i, 3) :
 [[ 0 -1]
 [ 1  0]]

matrix_power(i, 0):
 [[1 0]
 [0 1]]

matrix_power(i, -3):
 [[ 0.  1.]
 [-1.  0.]]
```

© 2000-2025, MetaQuotes Ltd.

Dot

두 벡터의 내적.

```
double vector::Dot(
  const vector&  b      // second vector
   );
```

매개 변수

b

[in] Vector.

반환값

Scalar.

참조

두 행렬의 내적은 행렬 곱 matrix::MatMul()입니다.

MQL5에서 벡터의 스칼라 곱과 관련한 간단한 알고리즘:

```
double VectorDot(const vector& vector_a, const vector& vector_b)
  {
   double dot=0;

   if(vector_a.Size()==vector_b.Size())
     {
      for(ulong i=0; i<vector_a.Size(); i++)
         dot+=vector_a[i]*vector_b[i];
     }

   return(dot);
  }
```

MQL5 예:

```
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

파이썬 예:

```
import numpy as np

a = [1, 0, 0, 1]
b = [4, 1, 2, 2]
print(np.dot(a, b))

>>> 6
```

© 2000-2025, MetaQuotes Ltd.

Kron

두 행렬, 행렬과 벡터, 벡터와 행렬 또는 두 벡터의 Kronecker 곱을 반환합니다.

```
matrix matrix::Kron(
  const matrix&  b      // second matrix
   );

matrix matrix::Kron(
  const vector&  b      // vector
   );

matrix vector::Kron(
  const matrix&  b      // matrix
   );

matrix vector::Kron(
  const vector&  b      // second vector
   );
```

매개 변수

b

[in] second matrix.

반환값

행렬.

참조

Kronecker 곱은 블록 행렬 곱셈이라고도 합니다.

MQL5의 두 행렬에 대한 Kronecker 곱의 간단한 알고리즘:

```
matrix MatrixKronecker(const matrix& matrix_a,const matrix& matrix_b)
  {
   ulong  M=matrix_a.Rows();
   ulong  N=matrix_a.Cols();
   ulong  P=matrix_b.Rows();
   ulong  Q=matrix_b.Cols();
   matrix matrix_c(M*P,N*Q);

   for(ulong m=0; m<M; m++)
      for(ulong n=0; n<N; n++)
         for(ulong p=0; p<P; p++)
            for(ulong q=0; q<Q; q++)
```

© 2000-2025, MetaQuotes Ltd.

```
                matrix_c[m*P+p][n*Q+q]=matrix_a[m][n] * matrix_b[p][q];

    return(matrix_c);
   }
```

MQL5 예:

```
   matrix a={{1,2,3},{4,5,6}};
   matrix b=matrix::Identity(2,2);
   vector v={1,2};

   Print(a.Kron(b));
   Print(a.Kron(v));

  /*
    [[1,0,2,0,3,0]
     [0,1,0,2,0,3]
     [4,0,5,0,6,0]
     [0,4,0,5,0,6]]

    [[1,2,2,4,3,6]
     [4,8,5,10,6,12]]
  */
```

파이썬 예:

```
import numpy as np

A = np.arange(1,7).reshape(2,3)
B = np.identity(2)
V = [1,2]
print(np.kron(A, B))
print("")
print(np.kron(A, V))

[[1. 0. 2. 0. 3. 0.]
 [0. 1. 0. 2. 0. 3.]
 [4. 0. 5. 0. 6. 0.]
 [0. 4. 0. 5. 0. 6.]]

[[ 1  2  2  4  3  6]
 [ 4  8  5 10  6 12]]
```

© 2000-2025, MetaQuotes Ltd.

Inner

두 행렬의 내적.

```
matrix matrix::Inner(
  const matrix&  b      // second matrix
   );
```

매개 변수

b

[in] Matrix.

반환값

행렬.

참조

두 벡터의 내적은 두 벡터 vector::Dot()의 내적입니다.

MQL5에서 두 행렬의 내적과 관련한 간단한 알고리즘:

```
bool MatrixInner(matrix& c, const matrix& a, const matrix& b)
  {
//--- 열의 수는 같아야 합니다.
   if(a.Cols()!=b.Cols())
      return(false);
//--- 결과 행렬의 크기는 각 행렬의 벡터 수에 따라 다릅니다.
   ulong  rows=a.Rows();
   ulong  cols=b.Rows();
   matrix result(rows,cols);
//---
   for(ulong i=0; i<rows; i++)
     {
      vector v1=a.Row(i);
      for(ulong j=0; j<cols; j++)
        {
         vector v2=b.Row(j);
         result[i][j]=v1.Dot(v2);
        }
     }
//---
   c=result;
   return(true);
  }
```

MQL5 예:

© 2000-2025, MetaQuotes Ltd.

```
   matrix a={{0,1,2},{3,4,5}};
   matrix b={{0,1,2},{3,4,5},{6,7,8}};
   matrix c=a.Inner(b);
   Print(c);
   matrix a1={{0,1,2}};
   matrix c1=a1.Inner(b);
   Print(c1);

  /*
  [[5,14,23]
  [14,50,86]]
  [[5,14,23]]
  */
```

파이썬 예:

```
import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))

import numpy as np

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
A1= np.arange(3)
print(np.inner(A, B))
print("");
print(np.inner(A1, B))
```

© 2000-2025, MetaQuotes Ltd.

Outer

두 행렬 또는 두 벡터의 외적을 계산합니다.

```
matrix matrix::Outer(
  const matrix&  b      // second matrix
   );

matrix vector::Outer(
  const vector&  b      // second vector
   );
```

매개 변수

b

[in] Matrix.

반환값

행렬.

참조

Kronecker 곱과 마찬가지로 외부 곱도 블록 행렬(및 벡터) 곱셈입니다.

MQL5에서 두 행렬의 외적에 대한 간단한 알고리즘:

```
matrix MatrixOuter(const matrix& matrix_a, const matrix& matrix_b)
  {
//--- 결과 행렬의 크기는 행렬 크기에 따라 다릅니다.
   ulong  rows=matrix_a.Rows()*matrix_a.Cols();
   ulong  cols=matrix_b.Rows()*matrix_b.Cols();
   matrix matrix_c(rows,cols);
   ulong  cols_a=matrix_a.Cols();
   ulong  cols_b=matrix_b.Cols();
//---
   for(ulong i=0; i<rows; i++)
     {
      ulong row_a=i/cols_a;
      ulong col_a=i%cols_a;
      for(ulong j=0; j<cols; j++)
        {
         ulong row_b=j/cols_b;
         ulong col_b=j%cols_b;
         matrix_c[i][j]=matrix_a[row_a][col_a] * matrix_b[row_b][col_b];
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(matrix_c);
  }
```

MQL5 예:

```
   vector vector_a={0,1,2,3,4,5};
   vector vector_b={0,1,2,3,4,5,6};
   Print("vector_a.Outer\n",vector_a.Outer(vector_b));
   Print("vector_a.Kron\n",vector_a.Kron(vector_b));

   matrix matrix_a={{0,1,2},{3,4,5}};
   matrix matrix_b={{0,1,2},{3,4,5},{6,7,8}};
   Print("matrix_a.Outer\n",matrix_a.Outer(matrix_b));
   Print("matrix_a.Kron\n",matrix_a.Kron(matrix_b));

  /*
   vector_a.Outer
   [[0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6]
    [0,2,4,6,8,10,12]
    [0,3,6,9,12,15,18]
    [0,4,8,12,16,20,24]
    [0,5,10,15,20,25,30]]
   vector_a.Kron
   [[0,0,0,0,0,0,0,0,1,2,3,4,5,6,0,2,4,6,8,10,12,0,3,6,9,12,15,18,0,4,8,12,16,20,24,0,
   matrix_a.Outer
   [[0,0,0,0,0,0,0,0,0]
    [0,1,2,3,4,5,6,7,8]
    [0,2,4,6,8,10,12,14,16]
    [0,3,6,9,12,15,18,21,24]
    [0,4,8,12,16,20,24,28,32]
    [0,5,10,15,20,25,30,35,40]]
   matrix_a.Kron
   [[0,0,0,0,1,2,0,2,4]
    [0,0,0,3,4,5,6,8,10]
    [0,0,0,6,7,8,12,14,16]
    [0,3,6,0,4,8,0,5,10]
    [9,12,15,12,16,20,15,20,25]
    [18,21,24,24,28,32,30,35,40]]
   */
```

파이썬 예:

```
import numpy as np

A = np.arange(6)
```

© 2000-2025, MetaQuotes Ltd.

```
B = np.arange(7)
print("np.outer")
print(np.outer(A, B))
print("np.kron")
print(np.kron(A, B))

A = np.arange(6).reshape(2, 3)
B = np.arange(9).reshape(3, 3)
print("np.outer")
print(np.outer(A, B))
print("np.kron")

np.outer
[[ 0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6]
 [ 0  2  4  6  8 10 12]
 [ 0  3  6  9 12 15 18]
 [ 0  4  8 12 16 20 24]
 [ 0  5 10 15 20 25 30]]
np.kron
[ 0  0  0  0  0  0  0  0  1  2  3  4  5  6  0  2  4  6  8 10 12  0  3  6
  9 12 15 18  0  4  8 12 16 20 24  0  5 10 15 20 25 30]
np.outer
[[ 0  0  0  0  0  0  0  0  0]
 [ 0  1  2  3  4  5  6  7  8]
 [ 0  2  4  6  8 10 12 14 16]
 [ 0  3  6  9 12 15 18 21 24]
 [ 0  4  8 12 16 20 24 28 32]
 [ 0  5 10 15 20 25 30 35 40]]
np.kron
[[ 0  0  0  0  1  2  0  2  4]
 [ 0  0  0  3  4  5  6  8 10]
 [ 0  0  0  6  7  8 12 14 16]
 [ 0  3  6  0  4  8  0  5 10]
 [ 9 12 15 12 16 20 15 20 25]
 [18 21 24 24 28 32 30 35 40]]
```

© 2000-2025, MetaQuotes Ltd.

CorrCoef

Pearson 상관 계수(선형 상관 계수)를 계산합니다.

```
matrix matrix::CorrCoef(
  const bool    rowvar=true  // 관측값의 행 또는 열 벡터
);

scalar vector::CorrCoef(
  const vector&  b           // 두번째 벡터
);
```

반환값

Pearson 적률 상관 계수.

참조

상관 계수의 범위는 [-1, 1]입니다.

부동 소수점 반올림으로 인해 결과 배열이 Hermitian이 아닐 수 있고 대각선 요소가 1이 아닐 수 있으며 요소가 부등식 abs(a) <= 1을 충족하지 않을 수 있습니다. 실제 및 허수 부분은 해당 상황을 개선하기 위해 간격 [-1, 1]로 잘리지만 복잡한 경우에는 별로 도움이 되지 않습니다.

MQL5를 사용하여 두 벡터의 상관 계수를 계산하는 간단한 알고리즘:

```
double VectorCorrelation(const vector& vector_x,const vector& vector_y)
  {
   ulong n=vector_x.Size()<vector_y.Size() ? vector_x.Size() : vector_y.Size();
   if(n<=1)
      return(0);

   ulong  i;
   double xmean=0;
   double ymean=0;
   for(i=0; i<n; i++)
     {
      if(!MathIsValidNumber(vector_x[i]))
         return(0);
      if(!MathIsValidNumber(vector_y[i]))
         return(0);
      xmean+=vector_x[i];
      ymean+=vector_y[i];
     }
   xmean/=(double)n;
   ymean/=(double)n;
```

© 2000-2025, MetaQuotes Ltd.

```
   double s=0;
   double xv=0;
   double yv=0;
   double t1=0;
   double t2=0;
//--- 계산
   s=0;
   for(i=0; i<n; i++)
     {
      t1=vector_x[i]-xmean;
      t2=vector_y[i]-ymean;
      xv+=t1*t1;
      yv+=t2*t2;
      s+=t1*t2;
     }
//--- 확인
   if(xv==0 || yv==0)
      return(0);
//--- 결과 반환
   return(s/(MathSqrt(xv)*MathSqrt(yv)));
  }
```

MQL5 예:

```
   vectorf vector_a={1,2,3,4,5};
   vectorf vector_b={0,1,0.5,2,2.5};
   Print("vectors correlation ",vector_a.CorrCoef(vector_b));
//---
   matrixf matrix_a={{1,2,3,4,5},
                     {0,1,0.5,2,2.5}};
   Print("matrix rows correlation\n",matrix_a.CorrCoef());
   matrixf matrix_a2=matrix_a.Transpose();
   Print("transposed matrix cols correlation\n",matrix_a2.CorrCoef(false));
   matrixf matrix_a3={{1.0f, 2.0f, 3.0f, 4.0f, 5.0f},
                      {0.0f, 1.0f, 0.5f, 2.0f, 2.5f},
                      {0.1f, 1.0f, 2.0f, 1.0f, 0.3f}};
   Print("rows correlation\n",matrix_a3.CorrCoef());
   Print("cols correlation\n",matrix_a3.CorrCoef(false));

  /*
   vectors correlation 0.9149913787841797
   matrix rows correlation
   [[1,0.91499138]
    [0.91499138,1]]
   transposed matrix cols correlation
   [[1,0.91499138]
    [0.91499138,1]]
   rows correlation
```

© 2000-2025, MetaQuotes Ltd.

```
   [[1,0.91499138,0.08474271]
    [0.91499138,1,-0.17123166]
    [0.08474271,-0.17123166,1]]
   cols correlation
   [[1,0.99587059,0.85375023,0.91129309,0.83773589]
    [0.99587059,1,0.80295509,0.94491106,0.88385159]
    [0.85375023,0.80295509,1,0.56362146,0.43088508]
    [0.91129309,0.94491106,0.56362146,1,0.98827404]
    [0.83773589,0.88385159,0.43088508,0.98827404,1]]
   */
```

파이썬 예:

```
import numpy as np
va=[1,2,3,4,5]
vb=[0,1,0.5,2,2.5]
print("vectors correlation")
print(np.corrcoef(va,vb))

ma=np.zeros((2,5))
ma[0,:]=va
ma[1,:]=vb
print("matrix rows correlation")
print(np.corrcoef(ma))
print("transposed matrix cols correlation")
print(np.corrcoef(np.transpose(ma),rowvar=False))
print("")

ma1=[[1,2,3,4,5],[0,1,0.5,2,2.5],[0.1,1,0.2,1,0.3]]
print("rows correlation\n",np.corrcoef(ma1))
print("cols correlation\n",np.corrcoef(ma1,rowvar=False))

transposed matrix cols correlation
[[1.         0.91499142]
 [0.91499142 1.        ]]

rows correlation
 [[1.         0.91499142 0.1424941 ]
 [0.91499142 1.         0.39657517]
 [0.1424941  0.39657517 1.        ]]
cols correlation
 [[1.         0.99587059 0.98226063 0.91129318 0.83773586]
 [0.99587059 1.         0.99522839 0.94491118 0.88385151]
 [0.98226063 0.99522839 1.         0.97234063 0.92527551]
 [0.91129318 0.94491118 0.97234063 1.         0.98827406]
 [0.83773586 0.88385151 0.92527551 0.98827406 1.        ]]
```

© 2000-2025, MetaQuotes Ltd.

Cov

공분산 행렬 계산.

```
matrix matrix::Cov(
  const bool     rowvar=true  // 관측값의 행 또는 cols 벡터
);

matrix matrix::Cov(
  const bool     rowvar,      // 관측값의 행 또는 cols 벡터
  const int      ddof         // 자유도의 델타 수준
);

matrix vector::Cov(
  const vector& b             // 두번째 벡터
);

matrix vector::Cov(
  const vector& b,            // 두번째 벡터
  const int      ddof         // 자유도의 델타 수준
);
```

매개 변수

rowwar

[in] rowvar가 true (기본값) 인 경우 각 행은 열의 관찰값과 함께 변수를 나타냅니다. 그렇지 않으면 관계가 전환됩니다: 각 열은 변수를 나타내며 행에는 관찰값이 포함되어 있습니다.

b

[in] 관측값의 두번째 벡터

ddof

[in] "자유도의 델타 수준": 계산에 사용 된 제수는 n -ddof이며, 여기서 n은 요소의 수를 나타냅니다. 기본값으로 ddof는 1.

주의

공분산 행렬 계산.

MQL5를 사용하여 두 벡터의 공분산 행렬을 계산하기위한 간단한 알고리즘 :

```
bool VectorCovariation(const vector& vector_a,const vector& vector_b,matrix& matrix_c)
  {
   int i,j;
   int m=2;
   int n=(int)(vector_a.Size()<vector_b.Size()?vector_a.Size():vector_b.Size());
//--- checks
   if(n<=1)
      return(false);
   for(i=0; i<n; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       if(!MathIsValidNumber(vector_a[i]))
          return(false);
       if(!MathIsValidNumber(vector_b[i]))
          return(false);
      }
//---
   matrix matrix_x(2,n);
   matrix_x.Row(vector_a,0);
   matrix_x.Row(vector_b,1);
   vector t=vector::Zeros(m);
//--- calculation
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         t[i]+=matrix_x[i][j]/double(n);
   for(i=0; i<m; i++)
      for(j=0; j<n; j++)
         matrix_x[i][j]-=t[i];
//--- syrk C=alpha*A^H*A+beta*C (beta=0이고 고려되지 않음)
   matrix_c=matrix::Zeros(m,m);
   for(i=0; i<m; i++)
     {
      for(j=0; j<n; j++)
        {
         double v=matrix_x[i][j]/(n-1);
         for(int i_=i; i_<m; i_++)
            matrix_c[i][i_]+=v*matrix_x[i_][j];
        }
     }
// --- 강제 대칭성
   for(i=0; i<m-1; i++)
      for(j=i+1; j<m; j++)
         matrix_c[j][i]=matrix_c[i][j];
//---
   return(true);
  }
```

MQL5 예:

```
   matrix matrix_a={{3,-2.1},{1.1,-1},{0.12,4.3}};
   Print("covariation cols\n",matrix_a.Cov(false));
   Print("covariation rows\n",matrix_a.Cov());

   vector vector_a=matrix_a.Col(0);
   vector vector_b=matrix_a.Col(1);
   Print("covariation vectors\n",vector_a.Cov(vector_b));

  /*
```

© 2000-2025, MetaQuotes Ltd.

```
   covariation cols
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
   covariation rows
   [[13.005,5.355,-10.659]
    [5.355,2.205,-4.389]
    [-10.659,-4.389,8.736199999999998]]
   covariation vectors
   [[2.144133333333333,-4.286]
    [-4.286,11.71]]
  */
```

파이썬 예:

```
import numpy as np
matrix_a=np.array([[3,-2.1],[1.1,-1],[0.12,4.3]])
matrix_c=np.cov(matrix_a,rowvar=False)
print("covariation cols\n",matrix_c)
matrix_c2=np.cov(matrix_a)
print("covariation rows\n",matrix_c2)

vector_a=matrix_a[:,0]
vector_b=matrix_a[:,1]
matrix_c3=np.cov(vector_a,vector_b)
print("covariation vectors\n",matrix_c3)

covariation cols
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
covariation rows
 [[ 13.005    5.355  -10.659 ]
 [  5.355    2.205   -4.389 ]
 [-10.659   -4.389    8.7362]]
covariation vectors
 [[ 2.14413333 -4.286     ]
 [-4.286      11.71      ]]
```

© 2000-2025, MetaQuotes Ltd.

Correlate

두 벡터의 상호 상관을 계산합니다.

```
vector vector::Correlate(
  const vector&           v,        // 벡터
  ENUM_VECTOR_CONVOLVE    mode      // 모드
   );
```

매개 변수

v

[in] 두번째 벡터

mode

【n】'mode' 매개변수는 선형 컨벌루션 계산 모드를 결정합니다. ENUM_VECTOR_CONVOLVE 열거에서 값

반환값

두 벡터의 상호 상관.

참조

'mode' 매개변수는 선형 컨벌루션 계산 모드를 결정합니다.

MQL5를 사용하여 두 벡터의 상관 계수를 계산하는 간단한 알고리즘:

```
vector VectorCrossCorrelationFull(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[n-i-1]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationSame(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
   vector c=vector::Zeros(size);
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorCrossCorrelationValid(const vector& a,const vector& b)
  {
   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[n-i-1]*a[i_-i];
        }
     }

   return(c);
  }
```

MQL5 예:

```
   vector a={1,2,3,4,5};
   vector b={0,1,0.5};

   Print("full\n",a.Correlate(b,VECTOR_CONVOLVE_FULL));
   Print("same\n",a.Correlate(b,VECTOR_CONVOLVE_SAME));
   Print("valid\n",a.Correlate(b,VECTOR_CONVOLVE_VALID));
   Print("full\n",b.Correlate(a,VECTOR_CONVOLVE_FULL));

  /*
   full
```

© 2000-2025, MetaQuotes Ltd.

```
   [0.5,2,3.5,5,6.5,5,0]
   same
   [2,3.5,5,6.5,5]
   valid
   [3.5,5,6.5]
   full
   [0,5,6.5,5,3.5,2,0.5]
  */
```

파이썬 예:

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.correlate(a,b,'full'))
print("same\n",np.correlate(a,b,'same'));
print("valid\n",np.correlate(a,b,'valid'));
print("full\n",np.correlate(b,a,'full'))

full
 [0.5 2.  3.5 5.  6.5 5.  0. ]
same
 [2.  3.5 5.  6.5 5. ]
valid
 [3.5 5.  6.5]
full
 [0.  5.  6.5 5.  3.5 2.  0.5]
```

© 2000-2025, MetaQuotes Ltd.

Convolve

두 벡터의 불연속 선형 컨벌루션을 반환

```
vector vector::Convolve(
  const vector&            v,        // 벡터
  ENUM_VECTOR_CONVOLVE     mode      // 모드
   );
```

매개 변수

v

[out] 두번째 벡터

mode

【n】'mode' 매개변수는 선형 컨벌루션 계산 모드를 결정합니다.ENUM_VECTOR_CONVOLVE.

반환값

두 벡터의 불연속 선형 컨볼루션입니다.

MQL5에서 두 벡터의 컨벌루션을 계산하기 위한 간단한 알고리즘:

```
vector VectorConvolutionFull(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionFull(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=m+n-1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
      for(int i_=i; i_<i+m; i_++)
         c[i_]+=b[i]*a[i_-i];

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionSame(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionSame(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n);
```

© 2000-2025, MetaQuotes Ltd.

```
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-size/2+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
vector VectorConvolutionValid(const vector& a,const vector& b)
  {
   if(a.Size()<b.Size())
      return(VectorConvolutionValid(b,a));

   int    m=(int)a.Size();
   int    n=(int)b.Size();
   int    size=MathMax(m,n)-MathMin(m,n)+1;
   vector c=vector::Zeros(size);

   for(int i=0; i<n; i++)
     {
      for(int i_=i; i_<i+m; i_++)
        {
         int k=i_-n+1;
         if(k>=0 && k<size)
            c[k]+=b[i]*a[i_-i];
        }
     }

   return(c);
  }
```

MQL5 예:

```
  vector a= {1, 2, 3, 4, 5};
  vector b= {0, 1, 0.5};

  Print("full\n", a.Convolve(b, VECTOR_CONVOLVE_FULL));
  Print("same\n", a.Convolve(b, VECTOR_CONVOLVE_SAME));
  Print("valid\n", a.Convolve(b, VECTOR_CONVOLVE_VALID));
```

© 2000-2025, MetaQuotes Ltd.

```
/*
 full
 [0,1,2.5,4,5.5,7,2.5]
 same
 [1,2.5,4,5.5,7]
 valid
 [2.5,4,5.5]
*/
```

파이썬 예:

```
import numpy as np
a=[1,2,3,4,5]
b=[0,1,0.5]

print("full\n",np.convolve(a,b,'full'))
print("same\n",np.convolve(a,b,'same'));
print("valid\n",np.convolve(a,b,'valid'));


full
 [0.  1.  2.5 4.  5.5 7.  2.5]
same
 [1.  2.5 4.  5.5 7. ]
valid
 [2.5 4.  5.5]
```

© 2000-2025, MetaQuotes Ltd.

행렬 변환

행렬 분해는 다음과 같은 경우에 사용할 수 있습니다:

- 선형 방정식 시스템을 풀 때 중간 단계로
- 행렬 반전
- 행렬식을 계산할 때
- 행렬의 고유값과 고유벡터를 찾을 때
- 행렬의 분석 함수를 계산할 때
- 최소자승법을 사용할 때
- 미분 방정식의 수치해에서

문제에 따라 각기 다른 행렬 분해 유형이 사용됩니다.

함수	액션
Cholesky	Cholesky 분해를 계산
Eig	정사각형 행렬의 고유값과 오른쪽 고유벡터를 계산
EigVals	일반 행렬의 고유값을 계산
LU	하부 삼각 행렬과 상부 삼각 행렬의 곱으로 행렬의 LU 분해
LUP	행 순열만 있는 LU 분해를 나타내는 부분 피벗을 사용한 LUP 분해: PA=LU
QR	행렬의 qr 인수분해를 계산
SVD	특이값 분해

© 2000-2025, MetaQuotes Ltd.

Cholesky

촐레스키 분해를 계산

```
bool matrix::Cholesky(
  matrix&  L      // 행렬
   );
```

매개 변수

L key

[out】하부 삼각 행렬.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

정사각 행렬 a의 촐레스키 분해 L * LH를 반환합니다. 여기서 L은 하부 삼각 행렬이고 .H는 켤레 전치 연산자(a가 실원소인 경우 일반 전치 연산자임)입니다. a는 에르미트(실원소이면 대칭)이고 양정행렬이여야 합니다. a가 Hermitian인지 여부를 확인하기 위한 검사는 수행되지 않습니다. 또한 a의 하부 삼각 요소와 대각선 요소만 사용됩니다. L만 실제로 반환됩니다.

예

```
matrix matrix_a= {{5.7998084, -2.1825367}, {-2.1825367, 9.85910595}};
matrix matrix_l;
Print("matrix_a\n", matrix_a);

matrix_a.Cholesky(matrix_l);
Print("matrix_l\n", matrix_l);
Print("check\n", matrix_l.MatMul(matrix_l.Transpose()));

/*
matrix_a
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
matrix_l
[[2.408279136645086,0]
 [-0.9062640068544704,3.006291985133859]]
check
[[5.7998084,-2.1825367]
 [-2.1825367,9.85910595]]
*/
```

© 2000-2025, MetaQuotes Ltd.

Eig

정사각 행렬의 고유값과 우고유벡터를 계산

```
bool matrix::Eig(
  matrix&  eigen_vectors,          // 고유 벡터의 행렬
  vector&  eigen_values            // 고유값의 벡터
   );
```

고유값과 고유벡터의 복소해

```
bool matrix::Eig(
  matrix<complex>& eigen_vectors, //  고유 벡터 행렬
  vector<complex>& eigen_values   // 고유값 벡터< /t5>
   );
```

매개 변수

eigen_vectors

【out】수직 고유 벡터의 행렬.

eigen_values

【out】고유값의 벡터.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

주의

고유값을 계산할 때 복잡한 해가 발견되면 계산이 중지되고 오류 코드가 4019(ERR_MATH_OVERFLOW)로 설정됩니다. Eig 메서드의 복소 오버로드를 사용하여 복소 공간에서 완전한 해를 구합니다.

복소 고유값의 허수 부분이 0인 경우 이는 실수 고유값입니다. 이는 아래의 예에서 볼 수 있습니다.

예:

```
void OnStart()
  {
   matrix matrix_a =
     {
        {-3.474589, 1.106384, -9.091977,-3.925227 },
        {-5.522139, 2.366887,-15.162351,-6.357512 },
        { 8.394926,-2.960067, 22.292115, 9.524129 },
        { 7.803242,-2.080287, 19.217706, 8.186645 }
     };

   matrix eigen_vectors;
   vector eigen_values;

   bool res=matrix_a.Eig(eigen_vectors, eigen_values);
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("res=",res,"  error=",GetLastError());
   Print("Eigen vectors:\n",eigen_vectors, "\nEigen Values:\n",eigen_values);

//--- A * v 정확성 확인 = lambda * v
   int vectors=0;
   for(ulong n=0; n<eigen_values.Size(); n++)
     {
      vector eigen_vector=eigen_vectors.Col(n);
      vector vector_d1   =eigen_vector*eigen_values[n];
      vector vector_d2   =matrix_a.MatMul(eigen_vector);

      ulong errors=vector_d1.Compare(vector_d2,1e-13);
      if(errors==0)
         vectors++;
     }
   Print("vectors=",vectors);

//--- 복소 해
   matrix<complex> eigen_vectors_c;
   vector<complex> eigen_values_c;
   ResetLastError();
   res=matrix_a.Eig(eigen_vectors_c,eigen_values_c);
   Print("res=",res,"  error=",GetLastError());
   Print("Eigen vectors:\n",eigen_vectors_c, "\nEigen Values:\n",eigen_values_c);

//--- A * v 정확성 확인 = lambda * v
   matrixc matrix_c;
   matrix_c.Assign(matrix_a);
   vectors=0;
   for(ulong n=0; n<eigen_values_c.Size(); n++)
     {
      vectorc eigen_vector_c=eigen_vectors_c.Col(n);
      vectorc vector_c1     =eigen_vector_c*eigen_values_c[n];
      vectorc vector_c2     =matrix_c.MatMul(eigen_vector_c);

      ulong errors=vector_c1.Compare(vector_c2,1e-13);
      if(errors==0)
         vectors++;
     }
   Print("vectors=",vectors);
  }
/* Result
   res=true  error=4019
  고유 벡터:
   [[0.2649667608713664]
    [0.4488818803991876]
    [-0.6527335897527492]
    [-0.5497604331807768]]
  고유값:
```

© 2000-2025, MetaQuotes Ltd.

```
   [28.94158645962942]
   vectors=1
   res=true  error=0
  고유 벡터:
   [[(0.2649667608713664,0),(0.2227392219204762,0.3745470492013296),(0.403285439089771
    [(0.4488818803991876,0),(-0.2613452530342438,-0.3685707727819327),(-0.396850612637
    [(-0.6527335897527492,0),(-0.3418479634708521,-0.3299830378162041),(-0.355302103111
    [(-0.5497604331807768,0),(0.523227993452828,0.3262508584080381),(0.351283559614366
  고유값:
   [(28.94158645962942,0),(0.01022501104810897,0.02954980822331488),(0.010225011048108
   vectors=3
*/
```

© 2000-2025, MetaQuotes Ltd.

EigVals

일반 행렬의 고유값을 계산

```
bool matrix::EigVals(
  vector&  eigen_values  // 고유값의 벡터
   );
```

매개 변수

eigen_values

[out] Vector of right eigenvalues.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

EigVals와 Eig의 유일한 차이점은 EigVals는 고유 벡터를 계산하지 않고 고유 값만 계산한다는 것입니다.

© 2000-2025, MetaQuotes Ltd.

LU

하위 삼각 행렬과 상위 삼각 행렬의 곱인 행렬의 LU 분해.

```
bool matrix::LU(
  matrix&  L,     // 하부 삼각 행렬
  matrix&  U     // 상부 삼각 행렬
   );
```

매개 변수

L key

[out] 하부 삼각 행렬.

U

[out] 상부 삼각 행렬.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

예

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u;
//--- LU decomposition
   matrix_a.LU(matrix_l,matrix_u);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
//--- A = L * U인지 체크
   Print("check\n",matrix_l.MatMul(matrix_u));


   /*
   matrix_l
   [[1,0,0,0]
    [5,1,0,0]
    [8,0.875,1,0]
    [11,1.25,0.5904761904761905,1]]
   matrix_u
   [[1,2,3,4]
    [0,-8,-9,-13]
    [0,0,-13.125,-10.625]
    [0,0,0,-17.47619047619047]]
   check
   [[1,2,3,4]
    [5,2,6,7]
```

© 2000-2025, MetaQuotes Ltd.

```
    [8,9,3,10]
    [11,12,14,4]]
  */
```

© 2000-2025, MetaQuotes Ltd.

LUP

행 순열만 있는 LU 분해를 나타내는 부분 피벗을 사용한 LUP 분해: PA=LU.

```
bool  LUP(
  matrix&  L,     // 하부 삼각 행렬
  matrix&  U,     // 상부 삼각 행렬
  matrix&  P     // 순열 행렬
   );
```

매개 변수

L key

[out】하부 삼각 행렬.

U

【but】상부 삼각 행렬.

P

[out] 순열 행렬

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

예

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u,matrix_p;
//--- LUP 분해
   matrix_a.LUP(matrix_l,matrix_u,matrix_p);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
   Print("matrix_p\n",matrix_p);
//--- P * A = L * U 인지 체크
   Print("P * A\n",matrix_p.MatMul(matrix_a));
   Print("L * U\n",matrix_l.MatMul(matrix_u));

   /*
   matrix_l
   [[1,0,0,0]
    [0.4545454545454545,1,0,0]
    [0.7272727272727273,-0.07894736842105282,1,0]
    [0.09090909090909091,-0.2631578947368421,-0.2262773722627738,1]]
   matrix_u
   [[11,12,14,4]
    [0,-3.454545454545454,-0.3636363636363633,5.181818181818182]
```

© 2000-2025, MetaQuotes Ltd.

```
   [0,0,-7.210526315789473,7.500000000000001]
   [0,0,0,6.697080291970803]]
  matrix_p
  [[0,0,0,1]
   [0,1,0,0]
   [0,0,1,0]
   [1,0,0,0]]
  P * A
  [[11,12,14,4]
   [5,2,6,7]
   [8,9,3,10]
   [1,2,3,4]]
  L * U
  [[11,12,14,4]
   [5,2,6,7]
   [8,9,3.000000000000001,10]
   [1,2,3,4]]
  */
```

© 2000-2025, MetaQuotes Ltd.

QR

행렬의 qr 인수분해를 계산

```
bool  QR(
  matrix&  Q,    // 정규 직교 열이 있는 행렬
  matrix&  R      // 상부 삼각 행렬
   );
```

매개 변수

Q

[out] 정규 직교 열이 있는 행렬. mode = 'complete'일 때 결과는 a가 실수/복소수인지 여부에 따라 직교/유니터리 행렬입니다. 이 경우 행렬식은 +/-1이 될 수 있습니다. 입력 배열의 차원 수가 2보다 큰 경우 위의 속성을 가진 행렬 스택이 반환됩니다.

R key

【but】상부 삼각 행렬.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

예

```
//---  A*x = b
  matrix A = {{0, 1}, {1, 1}, {1, 1}, {2, 1}};
  Print("A \n", A);
  vector b = {1, 2, 2, 3};
  Print("b \n", b);
//--- A = Q*R
  matrix q, r;
  A.QR(q, r);
  Print("q \n", q);
  Print("r \n", r);
  matrix qr=q.MatMul(r);
  Print("qr \n", qr);
  /*
  A
  [[0,1]
  [1,1]
  [1,1]
  [2,1]]
  b
  [1,2,2,3]
  q
  [[0.4082482904638631,-0.8164965809277259,-1.110223024625157e-16,-0.4082482904638631]
  [0.4625425214347352,-0.03745747856526496,0.7041241452319315,0.5374574785652647]
  [-0.5374574785652648,-0.03745747856526496,0.7041241452319316,-0.4625425214347352]
  [-0.5749149571305296,-0.5749149571305299,-0.09175170953613698,0.5749149571305296]]
```

© 2000-2025, MetaQuotes Ltd.

```
   r
   [[-1.224744871391589,-0.2415816237971962]
   [-1.22474487139159,-1.466326495188786]
   [1.224744871391589,1.316496580927726]
   [1.224744871391589,0.2415816237971961]]
   qr
   [[-1.110223024625157e-16,1]
   [1,0.9999999999999999]
   [1,1]
   [2,1]]
   */
```

© 2000-2025, MetaQuotes Ltd.

SVD

특이값 분해.

```
bool matrix::SVD(
  matrix&  U,                    // 유니터리 행렬
  matrix&  V,                    // 유니터리 행렬
  vector&  singular_values       // 특이값
   );
```

매개 변수

U

【아웃】left singular vectors로 구성된 m 차의 유니터리 행렬.

V

【아웃】right singular vectors로 구성된 n차 단일 행렬.

singular_values

[out] 특이값

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

예

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
//--- SVD 분해 실행
  matrix U, V;
  vector singular_values;
  b.SVD(U, V, singular_values);
  Print("U \n", U);
  Print("V \n", V);
  Print("singular_values = ", singular_values);

// 블록 체크
//--- U * singular diagonal * V = A
  matrix matrix_s;
  matrix_s.Diag(singular_values);
  Print("matrix_s \n", matrix_s);
  matrix matrix_vt=V.Transpose();
  Print("matrix_vt \n", matrix_vt);
  matrix matrix_usvt=(U.MatMul(matrix_s)).MatMul(matrix_vt);
  Print("matrix_usvt \n", matrix_usvt);
```

© 2000-2025, MetaQuotes Ltd.

```
  ulong errors=(int)b.Compare(matrix_usvt, 1e-9);
  double res=(errors==0);
  Print("errors=", errors);

//---- 또다른 체크
  matrix U_Ut=U.MatMul(U.Transpose());
  Print("U_Ut \n", U_Ut);
  Print("Ut_U \n", (U.Transpose()).MatMul(U));

  matrix vt_V=matrix_vt.MatMul(V);
  Print("vt_V \n", vt_V);
  Print("V_vt \n", V.MatMul(matrix_vt));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
   [-1,0,1]
   [2,3,4]]
  U
  [[-0.7071067811865474,0.5773502691896254,0.408248290463863]
   [-6.827109697437648e-17,0.5773502691896253,-0.8164965809277256]
   [0.7071067811865472,0.5773502691896255,0.4082482904638627]]
  V
  [[0.5773502691896258,-0.7071067811865474,-0.408248290463863]
   [0.5773502691896258,1.779939029415334e-16,0.8164965809277258]
   [0.5773502691896256,0.7071067811865474,-0.408248290463863]]
  singular_values = [7.348469228349533,2.449489742783175,3.277709923350408e-17]

  matrix_s
  [[7.348469228349533,0,0]
   [0,2.449489742783175,0]
   [0,0,3.277709923350408e-17]]
  matrix_vt
  [[0.5773502691896258,0.5773502691896258,0.5773502691896256]
   [-0.7071067811865474,1.779939029415334e-16,0.7071067811865474]
   [-0.408248290463863,0.8164965809277258,-0.408248290463863]]
  matrix_usvt
  [[-3.999999999999997,-2.999999999999999,-2]
   [-0.9999999999999981,-5.977974170712231e-17,0.9999999999999974]
   [2,2.999999999999999,3.999999999999996]]
  errors=0

  U_Ut
  [[0.9999999999999993,-1.665334536937735e-16,-1.665334536937735e-16]
   [-1.665334536937735e-16,0.9999999999999987,-5.551115123125783e-17]
   [-1.665334536937735e-16,-5.551115123125783e-17,0.9999999999999999]]
```

© 2000-2025, MetaQuotes Ltd.

```
  Ut_U
  [[0.9999999999999993,-5.551115123125783e-17,-1.110223024625157e-16]
   [-5.551115123125783e-17,0.9999999999999987,2.498001805406602e-16]
   [-1.110223024625157e-16,2.498001805406602e-16,0.999999999999999]]
  vt_V
  [[1,-5.551115123125783e-17,0]
   [-5.551115123125783e-17,0.9999999999999996,1.110223024625157e-16]
   [0,1.110223024625157e-16,0.9999999999999996]]
  V_vt
  [[0.9999999999999999,1.110223024625157e-16,1.942890293094024e-16]
   [1.110223024625157e-16,0.9999999999999998,1.665334536937735e-16]
   [1.942890293094024e-16,1.665334536937735e-16,0.9999999999999996]
  */
 }
```

© 2000-2025, MetaQuotes Ltd.

통계적 메서드

행렬 및 벡터의 기술 통계를 계산하는 메서드.

함수	액션
ArgMax	최대값의 인덱스를 반환
ArgMin	최소값의 인덱스를 반환
Max	행렬/벡터의 최대값 반환
Min	행렬/벡터의 최소값 반환
Ptp	행렬/벡터 또는 주어진 행렬 축의 값의 범위를 반환
Sum	주어진 축(여러 축)에 대해서도 수행할 수 있는 행렬/벡터 요소의 합을 반환합니다.
Prod	주어진 축에 대해서도 수행할 수 있는 행렬/벡터 요소의 곱을 반환합니다.
CumSum	주어진 축을 따라 있는 요소를 포함하여 행렬/벡터 요소의 누적 합계를 반환
CumProd	주어진 축을 따라 있는 요소를 포함하여 행렬/벡터 요소의 누적 곱을 반환
Percentile	지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 지정된 백분위수를 반환
Quantile	지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 지정된 분위수를 반환
Median	행렬/벡터 요소의 중앙값 계산
Mean	요소 값의 산술 평균 계산
Average	행렬/벡터 값의 가중 평균 계산
Std	주어진 축을 따라 행렬/벡터 요소 또는 요소 값의 표준 편차를 반환합니다.
Var	행렬/벡터 요소 값의 분산 계산
LinearRegression	계산된 선형 회귀 값으로 벡터/행렬을 계산합니다

© 2000-2025, MetaQuotes Ltd.

ArgMax

최대값의 인덱스를 반환합니다.

```
ulong vector::ArgMax();

ulong matrix::ArgMax();

vector matrix::ArgMax(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

최대값의 인덱스입니다.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.ArgMax(0);
   vector rows_max=matrix_a.ArgMax(1);
   ulong  matrix_max=matrix_a.ArgMax();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max index ",matrix_max,"  max value ",matrix_a.Flat(matrix_max));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[0,3,1]
   rows_max=[0,2,0,1]
   max index 5  max value 12.0
   */
```

© 2000-2025, MetaQuotes Ltd.

ArgMin

최소값의 인덱스를 반환합니다.

```
ulong vector::ArgMin();

ulong matrix::ArgMin();

vector matrix::ArgMin(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

최소값의 인덱스입니다.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.ArgMin(0);
   vector rows_min=matrix_a.ArgMin(1);
   ulong  matrix_min=matrix_a.ArgMin();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min index ",matrix_min,"  min value ",matrix_a.Flat(matrix_min));

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,0,0]
   rows_min=[2,0,2,0]
   min index 3  min value 1.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Max

행렬/벡터의 최대값을 반환합니다.

```
double vector::Max();

double matrix::Max();

vector matrix::Max(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

행렬/벡터의 최대값.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_max=matrix_a.Max(0);
   vector rows_max=matrix_a.Max(1);
   double matrix_max=matrix_a.Max();

   Print("cols_max=",cols_max);
   Print("rows_max=",rows_max);
   Print("max value ",matrix_max);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_max=[10,11,12]
   rows_max=[10,12,6,11]
   max value 12.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Min

행렬/벡터의 최소값을 반환합니다.

```
double vector::Min();

double matrix::Min();

vector matrix::Min(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

행렬/벡터의 최소값.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_min=matrix_a.Min(0);
   vector rows_min=matrix_a.Min(1);
   double matrix_min=matrix_a.Min();

   Print("cols_min=",cols_min);
   Print("rows_min=",rows_min);
   Print("min value ",matrix_min);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_min=[1,3,2]
   rows_min=[2,1,4,7]
   min value 1.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Ptp

행렬/벡터 또는 Max() -Min()에 해당하는 주어진 행렬 축의 값 범위를 반환합니다. Ptp -Peak to peak.

```
double vector::Ptp();

double matrix::Ptp();

vector matrix::Ptp(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 —가로축, 1 —세로축.

반환값

값의 범위(최대 -최소)가 있는 벡터 .

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_ptp=matrix_a.Ptp(0);
   vector rows_ptp=matrix_a.Ptp(1);
   double matrix_ptp=matrix_a.Ptp();

   Print("cols_ptp  ",cols_ptp);
   Print("rows_ptp  ",rows_ptp);
   Print("ptp value  ",matrix_ptp);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_ptp [9,8,10]
   rows_ptp [8,11,2,4]
   ptp value  11.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Sum

주어진 축(여러 축)에 대해서도 수행할 수 있는 행렬/벡터 요소의 합을 반환합니다.

```
double vector::Sum();

double matrix::Sum();

vector matrix::Sum(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

지정된 축(여러 축)에 대해 수행할 수도 있는 행렬/벡터 요소의 합입니다.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_sum=matrix_a.Sum(0);
   vector rows_sum=matrix_a.Sum(1);
   double matrix_sum=matrix_a.Sum();

   Print("cols_sum=",cols_sum);
   Print("rows_sum=",rows_sum);
   Print("sum value ",matrix_sum);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_sum=[24,27,27]
   rows_sum=[15,21,15,27]
   sum value 78.0
   */
```

© 2000-2025, MetaQuotes Ltd.

Prod

주어진 축에 대해서도 수행할 수 있는 행렬/벡터 요소의 곱을 반환합니다.

```
double vector::Prod(
  const double  initial=1      // 초기 승수
   );

double matrix::Prod(
  const double  initial=1      // 초기 승수
   );

vector matrix::Prod(
  const int     axis,          // axis
  const double  initial=1      // 초기 승수
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

initial=1

[in] 초기 승수.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vector cols_prod=matrix_a.Prod(0);
   vector rows_prod=matrix_a.Prod(1);
   double matrix_prod=matrix_a.Prod();

   Print("cols_prod=",cols_prod);
   cols_prod=matrix_a.Prod(0,0.1);
   Print("cols_prod=",cols_prod);
   Print("rows_prod=",rows_prod);
   Print("prod value ",matrix_prod);

  /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_prod=[420,1320,864]
   cols_prod=[42,132,86.40000000000001]
   rows_prod=[60,96,120,693]
```

© 2000-2025, MetaQuotes Ltd.

```
   prod value 479001600.0
*/
```

© 2000-2025, MetaQuotes Ltd.

CumSum

주어진 축을 따라 있는 요소를 포함하여 행렬/벡터 요소의 누적 합계를 반환합니다.

```
vector vector::CumSum();

vector matrix::CumSum();

matrix matrix::CumSum(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

주어진 축을 따라 요소의 누적 합계입니다.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumsum=matrix_a.CumSum(0);
   matrix rows_cumsum=matrix_a.CumSum(1);
   vector cumsum_values=matrix_a.CumSum();

   Print("cols_cumsum\n",cols_cumsum);
   Print("rows_cumsum\n",rows_cumsum);
   Print("cumsum values  ",cumsum_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumsum
   [[10,3,2]
    [11,11,14]
    [17,16,18]
    [24,27,27]]
   rows_cumsum
   [[10,13,15]
    [1,9,21]
    [6,11,15]
```

© 2000-2025, MetaQuotes Ltd.

```
    [7,18,27]]
   cumsum values  [10,13,15,16,24,36,42,47,51,58,69,78]
   */
```

© 2000-2025, MetaQuotes Ltd.

CumProd

주어진 축을 따라 있는 요소를 포함한 행렬/벡터 요소의 누적 곱을 반환합니다.

```
vector vector::CumProd();

vector matrix::CumProd();

matrix matrix::CumProd(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 각 열의 가로 축(예: 행 위), 1 — 각 행의 세로 축(예: 열 위)

반환값

주어진 축을 따른 요소의 누적 곱.

예

```
   matrix matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   matrix cols_cumprod=matrix_a.CumProd(0);
   matrix rows_cumprod=matrix_a.CumProd(1);
   vector cumprod_values=matrix_a.CumProd();

   Print("cols_cumprod\n",cols_cumprod);
   Print("rows_cumprod\n",rows_cumprod);
   Print("cumprod values  ",cumprod_values);

   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_cumprod
   [[10,3,2]
    [10,24,24]
    [60,120,96]
    [420,1320,864]]
   rows_cumprod
   [[10,30,60]
    [1,8,96]
    [6,30,120]
```

© 2000-2025, MetaQuotes Ltd.

```
 [7,77,693]]
cumprod values  [10,30,60,60,480,5760,34560,172800,691200,4838400,53222400,47900160
*/
```

© 2000-2025, MetaQuotes Ltd.

Percentile

지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 지정된 백분위수를 반환합니다.

```
double vector::Percentile(
  const int  percent      //
   );

double matrix::Percentile(
  const int  percent      //
   );

vector matrix::Percentile(
  const int  percent,     //
  const int  axis         // axis
   );
```

매개 변수

percent

【n】계산할 백분위수는 반드시 0에서 100 사이여야 합니다.

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

백분위수: 스칼라 또는 벡터.

참조

'percent' 매개변수의 유효한 값은 [0, 100] 범위입니다. 선형 알고리즘은 백분위수를 계산하는 데 사용됩니다. percentiles를 올바르게 계산하려면 순서를 정렬해야 합니다.

예

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_percentile=matrix_a.Percentile(50,0);
   vectorf rows_percentile=matrix_a.Percentile(50,1);
   float matrix_percentile=matrix_a.Percentile(50);

   Print("cols_percentile ",cols_percentile);
   Print("rows_percentile ",rows_percentile);
   Print("percentile value  ",matrix_percentile);


   /*
   matrix_a
```

© 2000-2025, MetaQuotes Ltd.

```
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   cols_percentile [5.5,6.5,7.5]
   rows_percentile [2,5,8,11]
   percentile value  6.5
   */
```

© 2000-2025, MetaQuotes Ltd.

Quantile

지정된 축을 따라 행렬/벡터 요소 또는 요소 값의 지정된 분위수를 반환합니다.

```
double vector::Quantile(
  const double quantile      // 분위수
   );

double matrix::Quantile(
  const double quantile      // 분위수
   );

vector matrix::Quantile(
  const double quantile,     // 분위수
  const int    axis          // axis
   );
```

매개 변수

quantile

【n】계산할 분위수는 반드시 0과 1 사이여야 합니다.

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

분위수: 스칼라 또는 벡터.

참조

'quantile' 매개변수는 [0, 1] 범위의 값을 사용합니다. 분위수를 계산하는 데 선형 알고리즘이 사용됩니다. 분위수를 올바르게 계산하려면 시퀀스를 정렬해야 합니다.

예

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_quantile=matrix_a.Quantile(0.5,0);
   vectorf rows_quantile=matrix_a.Quantile(0.5,1);
   float matrix_quantile=matrix_a.Quantile(0.5);

   Print("cols_quantile ",cols_quantile);
   Print("rows_quantile ",rows_quantile);
   Print("quantile value  ",matrix_quantile);

   /*
   matrix_a
   [[1,2,3]
```

© 2000-2025, MetaQuotes Ltd.

```
    [4,5,6]
    [7,8,9]
    [10,11,12]]
  cols_quantile [5.5,6.5,7.5]
  rows_quantile [2,5,8,11]
  quantile value  6.5
  */
```

© 2000-2025, MetaQuotes Ltd.

Median

행렬/벡터 요소의 중앙값(median)을 계산합니다.

```
double vector::Median();

double matrix::Median();

vector matrix::Median(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

중앙값: 스칼라 또는 벡터.

참조

중앙값은 배열/벡터 요소의 상위 절반과 요소의 하위 절반을 구분하는 중간 값입니다. Quantile(0.5) 및 Percentile(50)과 동일합니다. percentiles를 올바르게 계산하려면 순서를 정렬해야 합니다.

예

```
   matrixf matrix_a={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_median=matrix_a.Median(0);
   vectorf rows_median=matrix_a.Median(1);
   float matrix_median=matrix_a.Median();

   Print("cols_median ",cols_median);
   Print("rows_median ",rows_median);
   Print("median value  ",matrix_median);


   /*
   matrix_a
   [[1,2,3]
    [4,5,6]
    [7,8,9]
    [10,11,12]]
   cols_median [5.5,6.5,7.5]
   rows_median [2,5,8,11]
```

© 2000-2025, MetaQuotes Ltd.

```
   median value  6.5
   */
```

© 2000-2025, MetaQuotes Ltd.

Mean

요소 값의 산술 평균을 계산합니다.

```
double vector::Mean();

double matrix::Mean();

vector matrix::Mean(
  const int  axis      // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

요소 값의 산술 평균입니다.

예

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_mean=matrix_a.Mean(0);
   vectorf rows_mean=matrix_a.Mean(1);
   float matrix_mean=matrix_a.Mean();

   Print("cols_mean ",cols_mean);
   Print("rows_mean ",rows_mean);
   Print("mean value  ",matrix_mean);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_mean [6,6.75,6.75]
   rows_mean [5,7,5,9]
   mean value  6.5
   */
```

© 2000-2025, MetaQuotes Ltd.

Average

행렬/벡터 값의 가중 평균을 계산합니다.

```
double vector::Average(
  const vector& weigts      // 가중 벡터
   );

double matrix::Average(
  const matrix& weigts      // 가중 행렬
);

vector matrix::Average(
  const matrix& weigts,     // 가중 행렬
  const int     axis        // 축
   );
```

매개 변수

axis

[in] Axis. 0 — 가로축, 1 — 세로축.

반환값

산술 평균(Arithmetic mean): 스칼라 또는 벡터.

참조

가중치 행렬/벡터는 기본 행렬/벡터와 연결됩니다.

예

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   matrixf matrix_w=matrixf::Ones(4,3);
   Print("matrix_a\n",matrix_a);

   vectorf cols_average=matrix_a.Average(matrix_w,0);
   vectorf rows_average=matrix_a.Average(matrix_w,1);
   float matrix_average=matrix_a.Average(matrix_w);

   Print("cols_average ",cols_average);
   Print("rows_average ",rows_average);
   Print("average value  ",matrix_average);


   /*
   matrix_a
   [[10,3,2]
```

© 2000-2025, MetaQuotes Ltd.

```
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_average [6,6.75,6.75]
   rows_average [5,7,5,9]
   average value  6.5
   */ value  6.5
```

© 2000-2025, MetaQuotes Ltd.

Std

주어진 축을 따라 행렬/벡터 요소 또는 요소 값의 표준 편차를 반환합니다.

```
double vector::Std();

double vector::Std(
  const int  ddof      // 자유도의 델타 수준
);

double matrix::Std();

vector matrix::Std(
  const int  axis      // 축
);

vector matrix::Std(
  const int  axis,     // 축
  const int  ddof      // 자유도의 델타 수준
);
```

매개 변수

axis

[in] 축. 0 — 가로축, 1 — 세로축.

ddof

[in] "자유도의 델타 수준": 계산에 사용 된 제수는 n -ddof이며, 여기서 n은 요소의 수를 나타냅니다. 기본값은 ddof 0.

반환 값

Standard_deviation: 스칼라 혹은 벡터

주의

표준 편차는 mean에서 제곱 편차의 평균의 제곱근입니다. i. e., std = sqrt(mean(x)), where x = abs(a - a.mean())**2.

평균 제곱 편차는 일반적으로 x.sum () / (n -ddof)로 계산되며, 여기서 n = len (x).

ddof=0로 계산된 식은 때때로 "모집단 표준편차(population standard deviation)"라고 불립니다. ddof > 0 (보통 1을 사용)일 때 계산된 값은 때때로 "표본 표준편차(sample standard deviation)"라고 불립니다.

예

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);
```

© 2000-2025, MetaQuotes Ltd.

```
vectorf cols_std=matrix_a.Std(0);
vectorf rows_std=matrix_a.Std(1);
float matrix_std=matrix_a.Std();

Print("cols_std ",cols_std);
Print("rows_std ",rows_std);
Print("std value  ",matrix_std);


/*
matrix_a
[[10,3,2]
 [1,8,12]
 [6,5,4]
 [7,11,9]]
cols_std [3.2403703,3.0310888,3.9607449]
rows_std [3.5590262,4.5460606,0.81649661,1.6329932]
std value  3.452052593231201
*/
```

© 2000-2025, MetaQuotes Ltd.

Var

행렬/벡터 요소의 값의 분산을 계산합니다.

```
double vector::Var();

double vector::Var(
  const int  ddof      // 자유도의 델타 수준
);

double matrix::Var();

vector matrix::Var(
  const int  axis      // 축
);

vector matrix::Var(
  const int  axis,     // 축
  const int  ddof      // 자유도의 델타 수준
);
```

매개 변수

axis

[in] 축. 0 — 가로축, 1 — 세로축.

ddof

[in] "자유도의 델타 수준": 계산에 사용 된 제수는 n -ddof이며, 여기서 n은 요소의 수를 나타냅니다. 기본값은 ddof 0.

반환 값

분산: 스칼라 혹은 벡터

주의

분산은 mean 평균에서 제곱 편차의 평균, 즉 var = 평균 (x), 여기서 x = abs (a -a.mean ()) ** 2입니다.

mean 평균은 일반적으로 x.sum () / (n -ddof)로 계산되며, 여기서 n = len (x).

DDOF = 0을 사용한 표현은 때때로 "인구 분산"이라고합니다. DDOF> 0 (가장 일반적으로 사용되는 1)인 경우 결과 수량을 때때로 "샘플 분산"이라고합니다.

예

```
   matrixf matrix_a={{10,3,2},{1,8,12},{6,5,4},{7,11,9}};
   Print("matrix_a\n",matrix_a);

   vectorf cols_var=matrix_a.Var(0);
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf rows_var=matrix_a.Var(1);
   float matrix_var=matrix_a.Var();

   Print("cols_var ",cols_var);
   Print("rows_var ",rows_var);
   Print("var value  ",matrix_var);


   /*
   matrix_a
   [[10,3,2]
    [1,8,12]
    [6,5,4]
    [7,11,9]]
   cols_var [10.5,9.1875,15.6875]
   rows_var [12.666667,20.666666,0.66666669,2.6666667]
   var value  11.916666984558105
   */
```

© 2000-2025, MetaQuotes Ltd.

LinearRegression

계산된 선형 회귀 값으로 벡터/행렬을 계산합니다.

```
vector vector::LinearRegression();

matrix matrix::LinearRegression(
  ENUM_MATRIX_AXIS  axis=AXIS_NONE  // 회귀가 계산되는 축
   );
```

패러미터

축

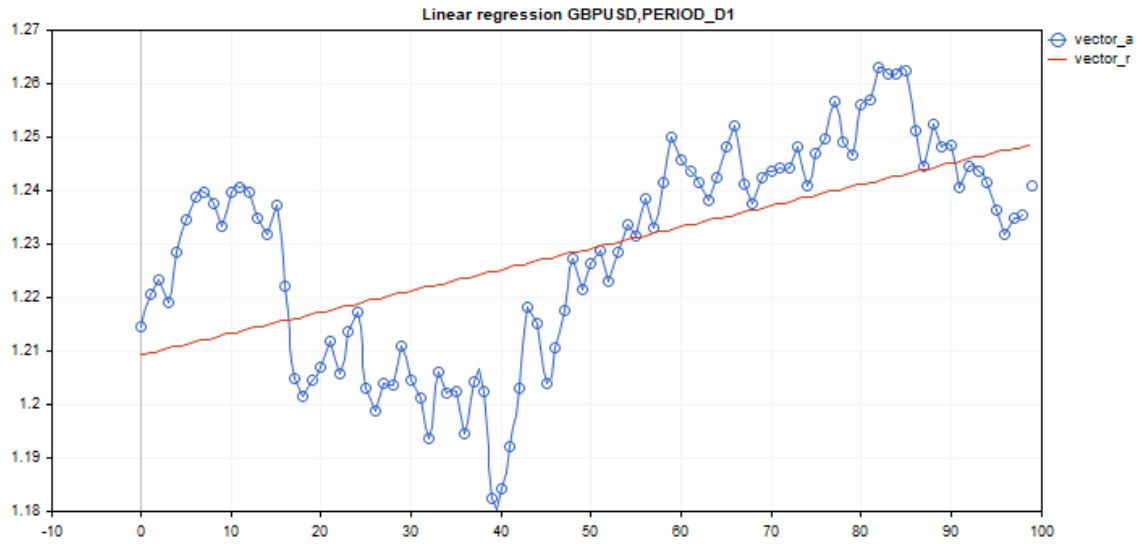
【n】회귀가 계산되는 축을 지정. ENUM_MATRIX_AXIS 열거형 값(AXIS_HORZ — 가로축, AXIS_VERT — 세로축).

반환 값

계산된 선형 회귀 값이 있는 벡터 또는 행렬.

주의

선형 회귀는 표준 회귀 방정식: y(x) = b + a * x + b를 사용하여 계산됩니다. 여기서 a는 선 기울기이고 b는 Y축 이동입니다.



예:

```
#include <Graphics\Graphic.mqh>

#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

/+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 프로그램 시작 함수 스크립트                                              |
/+------------------------------------------------------------------+
void OnStart()
  {
   vector vector_a;
   vector_a.CopyRates(_Symbol,_Period,COPY_RATES_CLOSE,1,100);
   vector vector_r=vector_a.LinearRegression();

//--- 차트 표시 끄기
   ChartSetInteger(0,CHART_SHOW,false);

//--- 그래프를 그리기 위한 배열
   double x[];
   double y1[];
   double y2[];
   ArrayResize(x,uint(vector_a.Size()));
   ArrayResize(y1,uint(vector_a.Size()));
   ArrayResize(y2,uint(vector_a.Size()));
   for(ulong i=0; i<vector_a.Size(); i++)
     {
      x[i]=(double)i;
      y1[i]=vector_a[i];
      y2[i]=vector_r[i];
     }

//--- 그래프 타이틀
   string title="Linear regression "+_Symbol+","+EnumToString(_Period);

   long   chart=0;
   string name="LinearRegression";

//--- 그래프 생성
   CGraphic graphic;
   graphic.Create(chart,name,0,0,0,GRAPH_WIDTH,GRAPH_HEIGHT);
   graphic.BackgroundMain(title);
   graphic.BackgroundMainSize(12);

//--- 활성화 함수 그래프
   CCurve *curvef=graphic.CurveAdd(x,y1,CURVE_POINTS_AND_LINES);
   curvef.Name("vector_a");
   curvef.LinesWidth(2);
   curvef.LinesSmooth(true);
   curvef.LinesSmoothTension(1);
   curvef.LinesSmoothStep(10);

//--- 활성화 함수의 미분
   CCurve *curved=graphic.CurveAdd(x,y2,CURVE_LINES);
   curved.Name("vector_r");
   curved.LinesWidth(2);
```

© 2000-2025, MetaQuotes Ltd.

```
   curved.LinesSmooth(true);
   curved.LinesSmoothTension(1);
   curved.LinesSmoothStep(10);
   graphic.CurvePlotAll();
   graphic.Update();

//--- 키보드 버튼을 누른것을 인식하는 무한 루프
   while(!IsStopped())
     {
//--- Esc 버튼을 눌러 프로그램을 종료
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
         break;
//--- 그래프 그림을 저장하려면 PdDn을 누르십시오
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0)
        {
         string file_names[];
         if(FileSelectDialog("Save Picture",NULL,"All files (*.*)|*.*",FSD_WRITE_FILE,
            continue;
         ChartScreenShot(0,file_names[0],GRAPH_WIDTH,GRAPH_HEIGHT);
        }
      Sleep(10);
     }

//--- 정리
   graphic.Destroy();
   ObjectDelete(chart,name);
   ChartSetInteger(0,CHART_SHOW,true);
  }
```

ENUM_MATRIX_AXIS

행렬에서 모든 통계 함수에서 축을 지정하기 위한 열거형

ID	설명
AXIS_NONE	축이 지정되지 않았습니다. 마치 벡터인 것처럼 모든 행렬 요소에 대해 계산이 수행됩니다.Flast 메서드를 참고).
AXIS_HORZ	수평축
AXIS_VERT	수직축

© 2000-2025, MetaQuotes Ltd.

기능 메서드

이러한 메서드를 사용하면 다음과 같은 행렬과 관련한 기능들을 수신할 수 있습니다:

- 행 수
- 열 수
- 노름
- 조건수
- 결정자
- 행렬 순위
- 트레이스
- 스펙트럼

함수	액션
Rows	행렬의 행 수를 반환
Cols	행렬의 열 수 반환
Size	벡터의 크기를 반환
Norm	행렬 또는 벡터 놈 반환
Cond	행렬의 조건수 계산
Det	정사각형 가역 행렬의 행렬식을 계산
SLogDet	행렬의 행렬식의 부호와 로그를 계산
Rank	가우스 방법을 사용하여 행렬 순위 반환
Trace	행렬의 대각선을 따라 합계를 반환
Spectrum	행렬곱 AT*A의 고유값 집합으로 행렬의 스펙트럼을 계산

© 2000-2025, MetaQuotes Ltd.

Rows

행렬의 행 수를 반환

```
ulong matrix::Rows()
```

반환값

정수.

예

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Rows()=", m.Rows());
Print("m.Cols()=", m.Cols());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Rows()=3
 m.Cols()=4
*/
```

© 2000-2025, MetaQuotes Ltd.

Cols

행렬의 열 수 반환

```
ulong matrix::Cols()
```

반환값

정수.

예

```
matrix m= {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}};
m.Reshape(3, 4);
Print("matrix m \n" , m);
Print("m.Cols()=", m.Cols());
Print("m.Rows()=", m.Rows());

/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 m.Cols()=4
 m.Rows()=3
*/
```

© 2000-2025, MetaQuotes Ltd.

Size

벡터의 사이즈를 반환합니다.

```
ulong vector::Size()
```

반환값

정수.

예

```
matrix m={{1,2,3,4,5,6,7,8,9,10,11,12}};
m.Reshape(3,4);
Print("matrix m\n",m);
vector v=m.Row(1);
Print("v.Size()=",v.Size());
Print("v=",v);


/*
 matrix m
 [[1,2,3,4]
  [5,6,7,8]
  [9,10,11,12]]
 v.Size()=4
 v=[5,6,7,8]
*/
```

© 2000-2025, MetaQuotes Ltd.

Norm

행렬 또는 벡터 노름을 반환

```
double vector::Norm(
  const ENUM_VECTOR_NORM  norm,     // 벡터 노름
  const int               norm_p=2  // VECTOR_NORM_P인 경우 p-norm의 수
   );

double matrix::Norm(
  const ENUM_MATRIX_NORM  norm      // 행렬 노름
   );
```

매개 변수

노름

[in] 노름 순서

반환값

행렬 또는 벡터 노름.

참조

- VECTOR_NORM_INF는 벡터 요소 중 최대 절대값입니다.
- VECTOR_NORM_MINUS_INF는 벡터의 최소 절대값입니다.
- VECTOR_NORM_P는 벡터의 P-노름입니다. norm_p=0이면 이는 0이 아닌 벡터 요소의 수입니다. norm_p=1은 벡터 요소의 절대값의 합입니다. norm_p=2는 벡터 요소 값의 제곱합의 제곱근입니다. P-norm은 음수가 될 수 있습니다.
- MATRIX_NORM_FROBENIUS는 행렬 요소 값의 제곱합의 제곱근입니다. Frobenius 노름과 벡터 P2 노름은 일치합니다.
- MATRIX_NORM_SPECTRAL은 매트릭스 스펙트럼의 최대값입니다.
- MATRIX_NORM_NUCLEAR는 행렬의 특이값의 합입니다.
- MATRIX_NORM_INF는 행렬의 수직 벡터 중 최대 벡터 p1-norm입니다. 행렬 inf-norm과 벡터 inf-norm은 일치합니다.
- MATRIX_NORM_MINUS_INF는 행렬의 수직 벡터 중 최소 벡터 p1-norm입니다.
- MATRIX_NORM_P1은 수평 행렬 벡터 중 최대 벡터 p1-norm입니다.
- MATRIX_NORM_MINUS_P1은 수평 행렬 벡터 중 최소 벡터 p1-norm입니다.
- MATRIX_NORM_P2는 행렬의 가장 높은 특이값입니다.
- MATRIX_NORM_MINUS_P2는 행렬의 가장 낮은 특이값입니다.

MQL5에서 벡터의 P-norm을 계산하기 위한 간단한 알고리즘:

```
double VectorNormP(const vector& v,int norm_value)
  {
   ulong  i;
   double norm=0.0;
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   switch(norm_value)
     {
      case 0 :
         for(i=0; i<v.Size(); i++)
            if(v[i]!=0)
               norm+=1.0;
         break;
      case 1 :
         for(i=0; i<v.Size(); i++)
            norm+=MathAbs(v[i]);
         break;
      case 2 :
         for(i=0; i<v.Size(); i++)
            norm+=v[i]*v[i];
         norm=MathSqrt(norm);
         break;
      default :
         for(i=0; i<v.Size(); i++)
            norm+=MathPow(MathAbs(v[i]),norm_value);
         norm=MathPow(norm,1.0/norm_value);
     }
//---
   return(norm);
  }
```

MQL5 예:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a=a-4;
  Print("matrix a \n", a);
  a.Reshape(3, 3);
  matrix b=a;
  Print("matrix b \n", b);
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_FROBENIUS)=", b.Norm(MATRIX_NORM_FROBENIUS));
  Print("b.Norm(MATRIX_NORM_INF)", b.Norm(MATRIX_NORM_INF));
  Print("b.Norm(MATRIX_NORM_MINUS_INF)", b.Norm(MATRIX_NORM_MINUS_INF));
  Print("b.Norm(MATRIX_NORM_P1)=)", b.Norm(MATRIX_NORM_P1));
  Print("b.Norm(MATRIX_NORM_MINUS_P1)=", b.Norm(MATRIX_NORM_MINUS_P1));
  Print("b.Norm(MATRIX_NORM_P2)=", b.Norm(MATRIX_NORM_P2));
  Print("b.Norm(MATRIX_NORM_MINUS_P2)=", b.Norm(MATRIX_NORM_MINUS_P2));

  /*
  matrix a
  [[-4,-3,-2,-1,0,1,2,3,4]]
  matrix b
  [[-4,-3,-2]
  [-1,0,1]
```

© 2000-2025, MetaQuotes Ltd.

```
   [2,3,4]]
   b.Norm(MATRIX_NORM_P2)=7.745966692414834
   b.Norm(MATRIX_NORM_FROBENIUS)=7.745966692414834
   b.Norm(MATRIX_NORM_INF)9.0
   b.Norm(MATRIX_NORM_MINUS_INF)2.0
   b.Norm(MATRIX_NORM_P1)=)7.0
   b.Norm(MATRIX_NORM_MINUS_P1)=6.0
   b.Norm(MATRIX_NORM_P2)=7.348469228349533
   b.Norm(MATRIX_NORM_MINUS_P2)=1.857033188519056e-16
   */
```

파이썬 예:

```
import numpy as np
from numpy import linalg as LA
a = np.arange(9) - 4
print("a \n",a)
b = a.reshape((3, 3))
print("b \n",b)
print("LA.norm(b)=",LA.norm(b))
print("LA.norm(b, 'fro')=",LA.norm(b, 'fro'))
print("LA.norm(b, np.inf)=",LA.norm(b, np.inf))
print("LA.norm(b, -np.inf)=",LA.norm(b, -np.inf))
print("LA.norm(b, 1)=",LA.norm(b, 1))
print("LA.norm(b, -1)=",LA.norm(b, -1))
print("LA.norm(b, 2)=",LA.norm(b, 2))
print("LA.norm(b, -2)=",LA.norm(b, -2))

a
 [-4 -3 -2 -1  0  1  2  3  4]
b
 [[-4 -3 -2]
 [-1  0  1]
 [ 2  3  4]]
LA.norm(b)= 7.745966692414834
LA.norm(b, 'fro')= 7.745966692414834
LA.norm(b, np.inf)= 9.0
LA.norm(b, -np.inf)= 2.0
LA.norm(b, 1)= 7.0
LA.norm(b, -1)= 6.0
LA.norm(b, 2)= 7.3484692283495345
LA.norm(b, -2)= 1.857033188519056e-16
```

© 2000-2025, MetaQuotes Ltd.

Cond

행렬의 조건수 계산

```
double matrix::Cond(
  const ENUM_MATRIX_NORM  norm      // 행렬 노름
   );
```

매개 변수

노름

[in] ENUM_MATRIX_NORM에서 노름의 순서

반환값

행렬의 조건수. 무한할 수 있습니다.

참조

x의 조건수는 x[1]의 역수의 x배의 노름으로 정의됩니다. 노름은 일반적인 L2 노름(제곱합의 루트) 또는 기타 여러 행렬의 노름 중 하나일 수 있습니다.

조건 umber는 행렬 A 놈의 곱과 그 역수와 같은 K 값입니다. 조건수가 높은 행렬을 나쁜 조건인 행렬이라고 합니다. 조건 번호가 낮은 것을 좋은 조건의 상태라고 합니다. 역행렬은 유사 역행을 이용하여 얻어지므로 행렬의 각짐과 비특이성 조건에 제약을 받지 않습니다.

스펙트럼 조건 번호는 예외입니다.

MQL5에서 스펙트럼 조건 수를 계산하기 위한 간단한 알고리즘:

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

MQL5 예:

```
matrix a= {{1, 0, -1}, {0, 1, 0}, { 1, 0, 1}};
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_FROBENIUS)=", a.Cond(MATRIX_NORM_FROBENIUS));
Print("a.Cond(MATRIX_NORM_INF)=", a.Cond(MATRIX_NORM_INF));
Print("a.Cond(MATRIX_NORM_MINUS_INF)=", a.Cond(MATRIX_NORM_MINUS_INF));
Print("a.Cond(MATRIX_NORM_P1)=)", a.Cond(MATRIX_NORM_P1));
Print("a.Cond(MATRIX_NORM_MINUS_P1)=", a.Cond(MATRIX_NORM_MINUS_P1));
Print("a.Cond(MATRIX_NORM_P2)=", a.Cond(MATRIX_NORM_P2));
Print("a.Cond(MATRIX_NORM_MINUS_P2)=", a.Cond(MATRIX_NORM_MINUS_P2));

/*
matrix a
[[1,0,-1]
[0,1,0]
[1,0,1]]
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_FROBENIUS)=3.162277660168379
a.Cond(MATRIX_NORM_INF)=2.0
a.Cond(MATRIX_NORM_MINUS_INF)=0.9999999999999997
a.Cond(MATRIX_NORM_P1)=)2.0
a.Cond(MATRIX_NORM_MINUS_P1)=0.9999999999999998
a.Cond(MATRIX_NORM_P2)=1.414213562373095
a.Cond(MATRIX_NORM_MINUS_P2)=0.7071067811865472
*/
```

파이썬 예:

```
import numpy as np
from numpy import linalg as LA
a = np.array([[1, 0, -1], [0, 1, 0], [1, 0, 1]])
print("a \n",a)
print("LA.cond(a)=",LA.cond(a))
print("LA.cond(a, 'fro')=",LA.cond(a, 'fro'))
print("LA.cond(a, np.inf)=",LA.cond(a, np.inf))
print("LA.cond(a, -np.inf)=",LA.cond(a, -np.inf))
print("LA.cond(a, 1)=",LA.cond(a, 1))
print("LA.cond(a, -1)=",LA.cond(a, -1))
print("LA.cond(a, 2)=",LA.cond(a, 2))
```

© 2000-2025, MetaQuotes Ltd.

```
print("LA.cond(a, -2)=",LA.cond(a, -2))

a
 [[ 1  0 -1]
 [ 0  1  0]
 [ 1  0  1]]
LA.cond(a)= 1.4142135623730951
LA.cond(a, 'fro')= 3.1622776601683795
LA.cond(a, np.inf)= 2.0
LA.cond(a, -np.inf)= 1.0
LA.cond(a, 1)= 2.0
LA.cond(a, -1)= 1.0
LA.cond(a, 2)= 1.4142135623730951
LA.cond(a, -2)= 0.7071067811865475
```

© 2000-2025, MetaQuotes Ltd.

Det

square 가역 행렬의 행렬식을 계산합니다.

```
double matrix::Det()
```

반환값

행렬식.

참조

2x2와 3x3 행렬식은 Sarrus 규칙에 따라 계산됩니다. d2=a11*a22-a12*a21; d3=a11*a22*a33+a12*a23*a31+a13*a21*a32-a13*a22*a31-a11*a23*a32-a12*a21*a33

행렬식은 행렬을 상부 삼각 형태로 축소하여 가우시안 방법으로 계산합니다. 상부 삼각 행렬의 행렬식은 주 대각선 요소의 곱과 같습니다.

하나 이상의 행렬 행 또는 열이 0이면 행렬식은 0입니다.

두 개 이상의 행렬 행 또는 열이 선형 종속인 경우 행렬식은 0입니다.

행렬의 행렬식은 고유값의 곱과 같습니다.

MQL5 예:

```
   matrix m={{1,2},{3,4}};
   double det=m.Det();
   Print("matrix m\n",m);
   Print("det(m)=",det);
   /*
   matrix m
   [[1,2]
    [3,4]]
   det(m)=-2.0
   */
```

파이썬 예:

```
import numpy as np

a = np.array([[1, 2], [3, 4]])
print('a \n',a)
print('nnp.linalg.det(a) \n',np.linalg.det(a))

a
 [[1 2]
 [3 4]]
```

© 2000-2025, MetaQuotes Ltd.

```
np.linalg.det(a)
 -2.0000000000000004
```

© 2000-2025, MetaQuotes Ltd.

SLogDet

행렬식의 부호와 로그를 계산합니다.

```
double matrix::SLogDet(
  int&  sign      // sign
   );
```

매개 변수

sign

【아웃】행렬식의 부호입니다. 부호가 짝수이면 행렬식은 양수입니다.

반환값

행렬식의 부호를 나타내는 숫자입니다.

참조

행렬식은 행렬을 상부 삼각 형태로 축소하여 가우시안 방법으로 계산합니다. 상부 삼각 행렬의 행렬식은 주 대각선 요소의 곱과 같습니다. 곱의 로그는 로그의 합과 같습니다. 따라서 행렬식을 계산할 때 오버플로가 발생하는 경우 SLogDet 방법을 사용할 수 있습니다.

부호가 짝수이면 행렬식은 양수입니다.

예

```
a = np.array([[1, 2], [3, 4]])  (sign, logdet) = np.linalg.slogdet(a)  (sign, logdet)
```

© 2000-2025, MetaQuotes Ltd.

Rank

가우스 방법을 사용하여 행렬 순위를 반환합니다.

```
int  Rank()
```

반환값

행렬의 순위.

참조

m개의 행과 n개의 열을 갖는 행렬 A의 행(또는 열)의 시스템의 순위는 선형적으로 독립인 행(또는 열)의 최대 개수입니다. 여러 행(열) 중 어느 것도 다른 항목에 대해 선형으로 표현할 수 없는 경우 선형 독립이라고 합니다. 행 시스템의 순위는 항상 열 시스템의 순위와 같습니다. 이 값을 행렬의 순위라고 합니다.

MQL5 예:

```
matrix a=matrix::Eye(4, 4);;
Print("matrix a \n", a);
Print("a.Rank()=", a.Rank());

matrix I=matrix::Eye(4, 4);
I[3, 3] = 0.;    // rank deficient matrix
Print("I \n", I);
Print("I.Rank()=", I.Rank());

matrix b=matrix::Ones(1, 4);
Print("b \n", b);
Print("b.Rank()=", b.Rank());;// 1 dimension - rank 1 unless all 0

matrix  zeros=matrix::Zeros(4, 1);
Print("zeros \n", zeros);
Print("zeros.Rank()=", zeros.Rank());

/*
matrix a
[[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[0,0,0,1]]
a.Rank()=4

I
[[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
```

© 2000-2025, MetaQuotes Ltd.

```
  [0,0,0,0]]
  I.Rank()=3

  b
  [[1,1,1,1]]
  b.Rank()=1

  zeros
  [[0]
  [0]
  [0]
  [0]]
  zeros.Rank()=0
  */
```

파이썬 예:

```
import numpy as np
from numpy.linalg import matrix_rank
a=(np.eye(4)) # Full rank matrix
print("a \n", a)
print("matrix_rank(a)=",matrix_rank(a))
I=np.eye(4)
I[-1,-1] = 0. # rank deficient matrix
print("I \n",I)
print("matrix_rank(I)=",matrix_rank(I))

b=np.ones((4,))
print("b \n",b)
print("matrix_rank(b)=",matrix_rank(b)) # 1 dimension - rank 1 unless all 0

zeros=np.zeros((4,))
print("zeroes \n",zeros)
print("matrix_rank(zeros)=",matrix_rank(zeros))

a
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
matrix_rank(a)= 4

I
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]]
matrix_rank(I)= 3
```

© 2000-2025, MetaQuotes Ltd.

```
b
 [1. 1. 1. 1.]
matrix_rank(b)= 1

zeroes
 [0. 0. 0. 0.]
matrix_rank(zeros)= 0
```

© 2000-2025, MetaQuotes Ltd.

Trace

행렬의 대각선을 따라 합계를 반환

```
double matrix::Trace()
```

반환값

대각선의 합계

참조

행렬의 대각합은 고유값의 합과 같습니다.

MQL5 예:

```
  matrix a= {{0, 1, 2, 3, 4, 5, 6, 7, 8}};
  a.Reshape(3, 3);
  Print("matrix a \n", a);
  Print("a.Trace() \n", a.Trace());

  /*
  matrix a
  [[0,1,2]
  [3,4,5]
  [6,7,8]]
  a.Trace()
  12.0
  */
```

파이썬 예:

```
a = np.arange(9).reshape((3,3))
print('a \n',a)
print('np.trace(a) \n',np.trace(a))

a
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

np.trace(a)
 12
```

© 2000-2025, MetaQuotes Ltd.

Spectrum

행렬곱 AT*A의 고유값 집합으로 행렬의 스펙트럼을 계산합니다.

```
vector matrix::Spectrum()
```

반환값

행렬 고유값의 벡터인 행렬의 스펙트럼.

예

```
double MatrixCondSpectral(matrix& a)
  {
   double norm=0.0;
   vector v=a.Spectrum();

   if(v.Size()>0)
     {
      double max_norm=v[0];
      double min_norm=v[0];
      for(ulong i=1; i<v.Size(); i++)
        {
         double real=MathAbs(v[i]);
         if(max_norm<real)
            max_norm=real;
         if(min_norm>real)
            min_norm=real;
        }
      max_norm=MathSqrt(max_norm);
      min_norm=MathSqrt(min_norm);
      if(min_norm>0.0)
         norm=max_norm/min_norm;
     }

   return(norm);
  }
```

© 2000-2025, MetaQuotes Ltd.

Matrix Classification

Раздел описывает набор функций для анализа структурных свойств матриц. Эти методы позволяют определить, является ли матрица симметричной, эрмитовой, диагональной, треугольной, трапециевидной, Хессенберга, бидиагональной или скалярной. Такая классификация упрощает выбор эффективных алгоритмов для линейной алгебры и оптимизацию вычислений

Функция	Действие
IsSymmetric	Check if a square matrix is symmetric
IsHermitian	Check if a square complex matrix is Hermitian
IsUpperTriangular	Check if a square matrix is upper triangular
IsLowerTriangular	Check if a square matrix is lower triangular
IsTrapezoidal	Check if a rectangular (not square) m-by-n matrix is upper or lower trapezoidal
IsUpperHessenberg	Check if a square matrix is upper Hessenberg matrix
IsLowerHessenberg	Check if a square matrix is lower Hessenberg matrix
IsTridiagonal	Check if a square matrix is tridiagonal
IsUpperBidiagonal	Check if a square matrix is upper bidiagonal
IsLowerBidiagonal	Check if a square matrix is lower bidiagonal
IsDiagonal	Check if a square matrix is diagonal
IsScalar	Check if a square matrix is scalar matrix

© 2000–2025, MetaQuotes Ltd.

IsSymmetric

Check if a square matrix is symmetric.

```
bool matrix::IsSymmetric();
```

Return Value

True if square matrix is symmetric.

Note

Zero matrix of n-by-n size is symmetric. Diagonal matrix is symmetric.

© 2000–2025, MetaQuotes Ltd.

IsHermitian

Check if a square complex matrix is Hermitian.

```
bool matrixc::IsHermitian();
```

Return Value

True if square complex matrix is Hermitian.

Note

Zero matrix of n-by-n size is Hermitian. Diagonal matrix is Hermitian only if all elements of the diagonal have an imaginary part equal to 0.

This method can be apply to the real (non-complex) matrix. In this case symmetry is checked.

© 2000–2025, MetaQuotes Ltd.

IsUpperTriangular

Check if a square matrix is upper triangular.

```
bool matrix::IsUpperTriangular();
```

Return Value

True if square matrix is upper triangular.

Note

Zero matrix of n-by-n size is triangular. Diagonal matrix is triangular. Bidiagonal matrix is triangular.

Check if lower triangular part under the main diagonal contains all zeros.

Upper triangular matrix

```
v  v  v  v  v  v
0  v  v  v  v  v
0  0  v  v  v  v
0  0  0  v  v  v
0  0  0  0  v  v
0  0  0  0  0  v
```

© 2000-2025, MetaQuotes Ltd.

IsLowerTriangular

Check if a square matrix is lower triangular.

```
bool matrix::IsLowerTriangular();
```

Return Value

True if square matrix is lower triangular.

Note

Zero matrix of n-by-n size is triangular. Diagonal matrix is triangular. Bidiagonal matrix is triangular.

Check if upper triangular part above the main diagonal contains all zeros.

Lower triangular matrix

```
v  0  0  0  0  0
v  v  0  0  0  0
v  v  v  0  0  0
v  v  v  v  0  0
v  v  v  v  v  0
v  v  v  v  v  v
```

© 2000-2025, MetaQuotes Ltd.

IsTrapezoidal

Check if a rectangular (not square) m-by-n matrix is upper or lower trapezoidal.

```
bool matrix::IsTrapezoidal(
  bool       is_upper      // upper or lover trapezoidal
   );
```

Parameters

is_upper

[out] Value specifies upper or lower trapezoidal matrix is recognized.

Return Value

True if matrix is trapezoidal.

Note

Zero matrix of m-by-n size is trapezoidal.

If m < n then check whether lower triangular part under the main diagonal contains all zeros. Zero matrix of 6-by-7 size is upper trapezoidal.

If m > n then check whether upper triangular part above the main diagonal contains all zeros. Zero matrix of 7-by-6 size is lower trapezoidal.

Trapezoidal matrices

```
   upper trapezoidal                lower trapezoidal

   v  v  v  v  v  v  v               v  0  0  0  0  0
   0  v  v  v  v  v  v               v  v  0  0  0  0
   0  0  v  v  v  v  v               v  v  v  0  0  0
   0  0  0  v  v  v  v               v  v  v  v  0  0
   0  0  0  0  v  v  v               v  v  v  v  v  0
   0  0  0  0  0  v  v               v  v  v  v  v  v
                                    v  v  v  v  v  v
```

© 2000–2025, MetaQuotes Ltd.

IsUpperHessenberg

Check if a square matrix is upper Hessenberg matrix.

```
bool matrix::IsUpperHessenberg();
```

Return Value

True if square matrix is upper Hessenberg matrix.

Note

Upper Hessenberg matrix contains all zeros under the subdiagonal.

Tridiagonal matrix is Hessenberg matrix. Upper triangular matrix is upper Hessenberg matrix.

Upper Hessenberg matrix

```
v  v  v  v  v  v
v  v  v  v  v  v
0  v  v  v  v  v
0  0  v  v  v  v
0  0  0  v  v  v
0  0  0  0  v  v
```

© 2000-2025, MetaQuotes Ltd.

IsLowerHessenberg

Check if a square matrix is lower Hessenberg matrix.

```
bool matrix::IsLowerHessenberg();
```

Return Value

True if square matrix is lower Hessenberg matrix.

Note

Lower Hessenberg matrix contains all zeros above the superdiagonal.

Tridiagonal matrix is Hessenberg matrix. Lower triangular matrix is lower Hessenberg matrix.

Upper Hessenberg matrix

```
v  v  0  0  0  0
v  v  v  0  0  0
v  v  v  v  0  0
v  v  v  v  v  0
v  v  v  v  v  v
v  v  v  v  v  v
```

© 2000-2025, MetaQuotes Ltd.

IsTridiagonal

Check if a square matrix is tridiagonal.

```
bool matrix::IsTridiagonal();
```

Return Value

True if square matrix is tridiagonal.

Note

Tridiagonal matrix contains all zeros under the subdiagonal and above the superdiagonal.

Bidiagonal matrix upper or lower is also tridiagonal.

Tridiagonal matrix

```
v  v  0  0  0  0
v  v  v  0  0  0
0  v  v  v  0  0
0  0  v  v  v  0
0  0  0  v  v  v
0  0  0  0  v  v
```

© 2000-2025, MetaQuotes Ltd.

IsUpperBidiagonal

Check if a square matrix is upper bidiagonal.

```
bool matrix::IsUpperBidiagonal();
```

Return Value

True if square matrix is upper bidiagonal.

Note

Upper bidiagonal matrix contains all zeros under the main diagonal and above the superdiagonal.

Diagonal matrix is also bidiagonal.

Upper bidiagonal matrix

```
v  v  0  0  0  0
0  v  v  0  0  0
0  0  v  v  0  0
0  0  0  v  v  0
0  0  0  0  v  v
0  0  0  0  0  v
```

© 2000-2025, MetaQuotes Ltd.

IsLowerBidiagonal

Check if a square matrix is lower bidiagonal.

```
bool matrix::IsLowerBidiagonal();
```

Return Value

True if square matrix is lower bidiagonal.

Note

Lower bidiagonal matrix contains all zeros under the subdiagonal and above the main diagonal.

Diagonal matrix is also bidiagonal.

Lower bidiagonal matrix

```
v  0  0  0  0  0
v  v  0  0  0  0
0  v  v  0  0  0
0  0  v  v  0  0
0  0  0  v  v  0
0  0  0  0  v  v
```

© 2000-2025, MetaQuotes Ltd.

IsDiagonal

Check if a square matrix is diagonal.

```
bool matrix::IsDiagonal();
```

Return Value

True if square matrix is diagonal.

Note

Diagonal matrix contains all zeros under and above the main diagonal.

Zero matrix of n-by-n size is diagonal.

Diagonal matrix

```
v  0  0  0  0  0
0  v  0  0  0  0
0  0  v  0  0  0
0  0  0  v  0  0
0  0  0  0  v  0
0  0  0  0  0  v
```

© 2000-2025, MetaQuotes Ltd.

IsScalar

Check if a square matrix is scalar matrix.

```
bool matrix::IsScalar();
```

Return Value

True if square matrix is scalar matrix.

Note

Scalar matrix is a diagonal matrix, where all diagonal elements are equal.

Zero matrix of n-by-n size is scalar matrix. Square identity matrix is scalar matrix.

Scalar matrix

```
I  0  0  0  0  0
0  I  0  0  0  0
0  0  I  0  0  0
0  0  0  I  0  0
0  0  0  0  I  0
0  0  0  0  0  I
```

© 2000-2025, MetaQuotes Ltd.

선형 방정식 시스템을 풀기 위한 매트릭스 메서드

선형 방정식 시스템을 풀고 역행렬을 계산하는 방법.

함수	액션
Solve	선형 행렬 방정식 또는 선형 대수 방정식의 시스템을 풉니다.
LstSq	선형 대수 방정식의 최소제곱해를 반환(비제곱 또는 축퇴 행렬의 경우).
Inv	Jordan-Gauss 메서드으로 정사각형 비축퇴 행렬의 (곱셈) 역을 계산합니다.
PInv	Moore-Penrose 방법으로 행렬의 유사 역행렬을 계산

© 2000-2025, MetaQuotes Ltd.

Solve

선형 행렬 방정식 또는 선형 대수 방정식의 시스템을 풉니다.

```
vector matrix::Solve(
  const 벡터b// 좌표 또는 '종속 변수' 값
   );
```

매개 변수

b

【in】좌표 또는 '종속 변수' 값. (Vector of free terms).

반환값

시스템 a * x = b에 대한 해가 있는 벡터.

참조

하나 이상의 행렬 행 또는 열이 0이면 시스템에 솔루션이 없습니다.

둘 이상의 행렬 행이나 열이 선형 종속인 경우 시스템에 솔루션이 없습니다.

예

```
//--- SLAE solution
   vector_x=matrix_a.Solve(vector_b);
//--- a * x = b인지 확인
   result_vector=matrix_a.MatMul(vector_x);
   errors=vector_b.Compare(result_vector,1e-12);
```

© 2000-2025, MetaQuotes Ltd.

LstSq

선형 대수 방정식의 최소 제곱해를 반환합니다(비제곱 또는 축퇴 행렬의 경우).

```
vector matrix::LstSq(
  const 벡터 b// 좌표 또는 '종속 변수' 값
   );
```

매개 변수

b

【n】좌표 또는 '종속 변수' 값. (Vector of free terms)

반환값

시스템 a*x=b에 대한 해가 있는 벡터. 이는 정확한 솔루션이 있는 시스템에만 해당됩니다.

예

```
   matrix a={{3, 2},
             {4,-5},
             {3, 3}};
   vector b={7,40,3};
//---
   vector x=a.LstSq(b);
//--- [5, -4]인지 확인
   Print("x=", x);
//--- [7, 40, 3]인지 확인
   vector b1=a.MatMul(x);
   Print("b1=",b1);

/*
  x=[5.000000000000002,-4]
  b1=[7.000000000000005,40.00000000000001,3.000000000000005]
*/
```

© 2000-2025, MetaQuotes Ltd.

Inv

Jordan-Gauss 방법으로 정사각형 가역 행렬의 곱셈 역행렬을 계산합니다.

```
matrix matrix::Inv()
```

반환값

Multiplicative inverse of the matrix.

참조

원래 행렬과 역행렬의 곱은 항등 행렬입니다.

적어도 하나의 행렬 행 또는 열이 0이면 역행렬을 얻을 수 없습니다.

둘 이상의 행렬 행 또는 열이 선형 종속인 경우 역행렬을 얻을 수 없습니다.

예

```
int TestInverse(const int size_m)
  {
   int    i,j,errors=0;
   matrix matrix_a(size_m,size_m);
//--- 정사각 행렬 채우기
   MatrixTestFirst(matrix_a);
//--- 마이크로초가 측정됩니다.
   ulong t1=GetMicrosecondCount();
//--- 역행렬 얻기
   matrix inverse=matrix_a.Inv();
//--- 측정
   ulong t2=GetMicrosecondCount();
//--- 정확성 확인
   matrix identity=matrix_a.MatMul(inverse);
//---
   for(i=0; i<size_m; i++)
     {
      for(j=0; j<size_m; j++)
        {
         double value;
//--- 대각선을 따라야 합니다.
         if(i==j)
            value=1.0;
         else
            value=0.0;
         if(MathClassify(identity[i][j])>FP_ZERO)
            errors++;
         else
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             if(identity[i][j]!=value)
               {
                double diff=MathAbs(identity[i][j]-value);
//--- 곱셈과 나눗셈이 너무 많아 검사 정확도를 낮춥니다.
                if(diff>1e-9)
                   errors++;
               }
            }
         }
      }
//---
   double elapsed_time=double(t2-t1)/1000.0;
   printf("Inversion of matrix %d x %d  %s  errors=%d  time=%.3f ms",size_m,size_m,er
   return(errors);
  }
```

© 2000-2025, MetaQuotes Ltd.

PInv

Moore-Penrose 메서드으로 행렬의 유사 역행렬을 계산합니다.

```
matrix matrix::PInv()
```

반환값

행렬의 유사 역행렬.

예

```
int TestPseudoInverse(const int size_m, const int size_k)
  {
   matrix matrix_a(size_m,size_k);
   matrix matrix_inverted(size_k,size_m);
   matrix matrix_temp;
   matrix matrix_a2;
//--- 행렬 채우기
   MatrixTestFirst(matrix_a);
//--- invert
   matrix_inverted=matrix_a.PInv();
//--- 정확성 확인
   int errors=0;
//--- A * A+ * A = A   (A+ is a pseudo-inverse of A)
   matrix_temp=matrix_a.MatMul(matrix_inverted);
   matrix_a2=matrix_temp.MatMul(matrix_a);
   errors=(int)matrix_a.CompareByDigits(matrix_a2,10);

   printf("PseudoInversion %s matrix_size  %d x %d  errors=%d",errors==0?"passed":"fai
//---
   return(errors);
  }
```

© 2000-2025, MetaQuotes Ltd.

머신 러닝

이러한 메서드들은 기계 학습에서 사용됩니다.

신경망 활성화 함수는 입력의 가중 합에 따라 뉴런의 출력 값을 결정합니다. 어떠한 활성화 함수를 선택할지 여부는 신경망 성능에 큰 영향을 미칩니다. 다른 모델 파트(레이어)는 다른 활성화 함수를 사용할 수 있습니다.

알려진 모든 활성화 기능 외에도 MQL5는 드것들의 파생 기능도 제공합니다. 도함수를 사용하면 학습에서 받은 오류를 기반으로 모델 매개변수를 효율적으로 업데이트할 수 있습니다.

신경망은 학습에서 오류를 최소화하는 알고리즘을 찾는 것을 목표로 하며 손실 함수를 사용합니다. 손실 함수의 값은 모델이 예측한 값이 실제 값과 얼마나 다른지를 나타냅니다. 문제에 따라 다른 손실 함수가 사용됩니다. 예를 들어 평균 제곱 오차(MSE)는 회귀 문제에 사용되며 Binary Cross-Entropy(기원전)는 이진 분류 목적으로 사용됩니다.

함수	액션
Activation	활성화 함수 값을 계산하고 전달된 벡터/행렬에 씁니다
Derivative	활성화 함수 도함수 값을 계산하고 이들을 전달된 벡터/행렬에 씁니다
Loss	손실 함수 값을 계산하고 전달된 벡터/행렬에 씁니다
LossGradient	손실 함수 기울기의 벡터 또는 행렬을 계산합니다
RegressionMetric	지정된 데이터 배열에 구성된 회귀선의 편차 오류로 회귀 메트릭을 계산
ConfusionMatrix	혼동 행렬 계산. 이 메서드는 예측값의 벡터에 적용됩니다
ConfusionMatrixMultilabel	각 레이블에 대한 혼동 행렬 계산. 이 메서드는 예측값의 벡터에 적용됩니다
ClassificationMetric	실제 데이터와 비교하여 예측 데이터의 품질을 평가하기 위해 분류 지표를 계산합니다. 이 방법은 예측 값의 벡터에 적용됩니다
ClassificationScore	실제 데이터와 비교하여 예측 데이터의 품질을 평가하기 위해 분류 메트릭을 계산
PrecisionRecall	값들을 계산하여 정밀도-재현율 곡선(precision-recall curve)을 구성합니다. ClassificationScore와 마찬가지로 이 메서드는 참값의 벡터에 적용됩니다
ReceiverOperatingCharacteristic	Receiver Operating Characteristic (ROC) 곡선을 구성하기 위한 값을 계산합니다. ClassificationScore와 마찬가지로 이 메서드는 참값의 벡터에 적용됩니다

예

이 예는 행렬 연산을 사용하여 모델을 훈련하는 방법을 보여줍니다. 모델은 (a + b + c)^2 / (a^2 + b^2 + c^2) 함수에 대해 학습됩니다. a, b 및 c가 서로 다른 열에 포함된 초기 데이터 매트릭스를 입력합니다. 함수 결과는 모델 출력에서 얻습니다.

© 2000-2025, MetaQuotes Ltd.

```
matrix weights1, weights2, weights3;                // 가중치의 행렬
matrix output1, output2, result;                    // 신경층 출력의 행렬
input int layer1 = 200;                             // 첫 번째 은닉층의 크기
input int layer2 = 200;                             // 두 번째 은닉층의 크기
input int Epochs = 20000;                           // 훈련 에포크의 수
input double lr = 3e-6;                             // 학습률
input ENUM_ACTIVATION_FUNCTION ac_func = AF_SWISH; // 활성화 함수
//+------------------------------------------------------------------+
//| Script start function                                            |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int train = 1000;    // 샘플 사이즈 훈련
   int test = 10;       // 샘플 사이즈 테스트
   matrix m_data, m_target;
//--- 훈련 샘플 생성
   if(!CreateData(m_data, m_target, train))
      return;
//--- 모델 훈련
   if(!Train(m_data, m_target, Epochs))
      return;
//--- 테스트 샘플 생성
   if(!CreateData(m_data, m_target, test))
      return;
//--- 모델 테스트
   Test(m_data, m_target);
  }
//+------------------------------------------------------------------+
//| Sample generation method                                         |
//+------------------------------------------------------------------+
bool CreateData(matrix &data, matrix &target, const int count)
  {
//--- 초기 데이터 및 결과 행렬 초기화
   if(!data.Init(count, 3) || !target.Init(count, 1))
      return false;
//--- 임의의 값으로 초기 데이터 행렬 채우기
   data.Random(-10, 10);
//--- 학습 샘플의 목표 값을 계산
   vector X1 = MathPow(data.Col(0) + data.Col(1) + data.Col(1), 2);
   vector X2 = MathPow(data.Col(0), 2) + MathPow(data.Col(1), 2) + MathPow(data.Col(2)
   if(!target.Col(X1 / X2, 0))
      return false;
//--- 결과 반환
   return true;
  }
//+------------------------------------------------------------------+
//| Model training method                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool Train(matrix &data, matrix &target, const int epochs = 10000)
  {
//--- 모델 생성
   if(!CreateNet())
      return false;
//--- 모델 훈련
   for(int ep = 0; ep < epochs; ep++)
     {
//--- 피드포워드 패스
      if(!FeedForward(data))
         return false;
      PrintFormat("Epoch %d, loss %.5f", ep, result.Loss(target, LOSS_MSE));
//--- 가중치 행렬의 역전파 및 업데이트
      if(!Backprop(data, target))
         return false;
     }
//--- 결과 반환
   return true;
  }
//+------------------------------------------------------------------+
//| Model creation method                                            |
//+------------------------------------------------------------------+
bool CreateNet()
  {
//--- 가중치 행렬 초기화
   if(!weights1.Init(4, layer1) || !weights2.Init(layer1 + 1, layer2) || !weights3.Ini
      return false;
//--- 가중치 행렬을 임의의 값으로 채웁니다.
   weights1.Random(-0.1, 0.1);
   weights2.Random(-0.1, 0.1);
   weights3.Random(-0.1, 0.1);
//--- 결과 반환
   return true;
  }
//+------------------------------------------------------------------+
//| Feedforward method                                               |
//+------------------------------------------------------------------+
bool FeedForward(matrix &data)
  {
//--- 초기 데이터 크기 확인
   if(data.Cols() != weights1.Rows() - 1)
      return false;
//--- 첫 번째 신경 계층을 계산합니다.
   matrix temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   output1 = temp.MatMul(weights1);
//--- 활성화 함수 계산
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!output1.Activation(temp, ac_func))
      return false;
//--- 두 번째 신경층을 계산합니다.
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   output2 = temp.MatMul(weights2);
//--- 활성화 함수 계산
   if(!output2.Activation(temp, ac_func))
      return false;
//--- 세 번째 신경층을 계산합니다.
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   result = temp.MatMul(weights3);
//--- 결과 반환
   return true;
  }
//+------------------------------------------------------------------+
//| Backpropagation method                                           |
//+------------------------------------------------------------------+
bool Backprop(matrix &data, matrix &target)
  {
//--- 대상 값의 행렬 크기 확인
   if(target.Rows() != result.Rows() ||
      target.Cols() != result.Cols())
      return false;
//--- 목표에서 계산된 값의 편차를 결정합니다.
   matrix loss = (target - result) * 2;
//--- 그래디언트를 이전 레이어로 전파
   matrix gradient = loss.MatMul(weights3.Transpose());
//--- 마지막 레이어의 가중 행렬 업데이트
   matrix temp;
   if(!output2.Activation(temp, ac_func))
      return false;
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   weights3 = weights3 + temp.Transpose().MatMul(loss) * lr;
//--- 활성화 함수의 도함수로 오류 기울기를 조정합니다.
   if(!output2.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- 기울기를 하위 레이어로 전파
   gradient = loss.MatMul(weights2.Transpose());
//--- 두 번째 은닉층의 가중치 행렬 업데이트
   if(!output1.Activation(temp, ac_func))
```

© 2000-2025, MetaQuotes Ltd.

```
      return false;
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   weights2 = weights2 + temp.Transpose().MatMul(loss) * lr;
//--- 활성화 함수의 도함수로 오류 기울기를 조정합니다.
   if(!output1.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- 첫 번째 은닉층의 가중치 행렬을 업데이트합니다.
   temp = data;
   if(!temp.Resize(temp.Rows(), weights1.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights1.Rows() - 1))
      return false;
   weights1 = weights1 + temp.Transpose().MatMul(loss) * lr;
//--- 결과 반환
   return true;
  }
//+------------------------------------------------------------------+
//| Model testing method                                             |
//+------------------------------------------------------------------+
bool Test(matrix &data, matrix &target)
  {
//--- 테스트 데이터에서 피드포워드
   if(!FeedForward(data))
      return false;
//--- 모델 계산 결과 및 실제 값 기록
   PrintFormat("Test loss %.5f", result.Loss(target, LOSS_MSE));
   ulong total = data.Rows();
   for(ulong i = 0; i < total; i++)
      PrintFormat("(%.2f + %.2f + %.2f)^2 / (%.2f^2 + %.2f^2 + %.2f^2) =  Net %.2f, Ta
                  data[i, 0], data[i, 1], data[i, 2], result[i, 0], target[i, 0]);
//--- 결과 반환
   return true;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

Activation

활성화 함수 값을 계산하고 전달된 벡터/행렬에 그 값을 씁니다.

```
bool vector::Activation(
  vector&                    vect_out,      // 값을 얻기 위한 벡터
  ENUM_ACTIVATION_FUNCTION   activation,    // 활성화 함수
   ...                                      // 추가 매개변수
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // 값을 얻기 위한 행렬
  ENUM_ACTIVATION_FUNCTION   activation     // 활성화 함수
   );


bool matrix::Activation(
  matrix&                    matrix_out,    // 값을 얻기 위한 행렬
  ENUM_ACTIVATION_FUNCTION   activation,    // 활성화 함수
  ENUM_MATRIX_AXIS           axis,          // 축
   ...                                      // 추가 매개변수
   );
```

매개 변수

vect_out/matrix_out

【but】활성화 함수의 계산된 값을 얻기 위한 벡터 또는 행렬.

activation

[in] ENUM_ACTIVATION_FUNCTION 열거로 부터의 활성화 함수.

axis

[in]ENUM_MATRIX_AXIS 열거형 값(AXIS_HORZ — 가로축, AXIS_VERT — 세로축).

...

【n】일부 활성화 함수에 필요한 추가 매개변수. 매개변수를 지정하지 않으면 기본값이 사용됩니다.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

추가 매개변수

일부 활성화 함수는 추가 매개변수를 허용합니다. 매개변수를 지정하지 않으면 기본값이 사용됩니다.

```
   AF_ELU   (Exponential Linear Unit)
```

© 2000-2025, MetaQuotes Ltd.

```
  double alpha=1.0

Activation function: if(x>=0) f(x) = x
                     else f(x) = alpha * (exp(x)-1)


AF_LINEAR
  double alpha=1.0
  double beta=0.0

Activation function: f(x) = alpha*x + beta


AF_LRELU   (Leaky REctified Linear Unit)
  double alpha=0.3

Activation function: if(x>=0) f(x)=x
                     else f(x) = alpha*x


AF_RELU  (REctified Linear Unit)
  double alpha=0.0
  double max_value=0.0
  double treshold=0.0

Activation function: if(alpha==0) f(x) = max(x,0)
                     else if(x>max_value) f(x) = x
                     else f(x) = alpha*(x - treshold)


AF_SWISH
  double beta=1.0

Activation function: f(x) = x / (1+exp(-x*beta))


AF_TRELU   (Thresholded REctified Linear Unit)
  double theta=1.0

Activation function: if(x>theta) f(x) = x
                     else f(x) = 0


AF_PRELU   (Parametric REctified Linear Unit)
  double alpha[] - learned array of coeefficients

Activation function: if(x[i]>=0) f(x)[i] = x[i]
                     else f(x)[i] = alpha[i] * x[i]
```

© 2000-2025, MetaQuotes Ltd.

참조

인공 신경망에서 뉴런의 활성화 함수는 입력 신호 또는 입력 신호의 집합으로 정의되는 출력 신호를 결정합니다. 어떠한 활성화 함수를 선택할지 여부는 신경망 성능에 큰 영향을 미칩니다. 다른 모델 파트(레이어)는 다른 활성화 함수를 사용할 수 있습니다.

추가 매개변수 사용의 예:

```
vector x={0.1, 0.4, 0.9, 2.0, -5.0, 0.0, -0.1};
vector y;

x.Activation(y,AF_ELU);
Print(y);
x.Activation(y,AF_ELU,2.0);
Print(y);

Print("");
x.Activation(y,AF_LINEAR);
Print(y);
x.Activation(y,AF_LINEAR,2.0);
Print(y);
x.Activation(y,AF_LINEAR,2.0,5.0);
Print(y);

Print("");
x.Activation(y,AF_LRELU);
Print(y);
x.Activation(y,AF_LRELU,1.0);
Print(y);
x.Activation(y,AF_LRELU,0.1);
Print(y);

Print("");
x.Activation(y,AF_RELU);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5);
Print(y);
x.Activation(y,AF_RELU,2.0,0.5,1.0);
Print(y);

Print("");
x.Activation(y,AF_SWISH);
Print(y);
x.Activation(y,AF_SWISH,2.0);
Print(y);
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("");
   x.Activation(y,AF_TRELU);
   Print(y);
   x.Activation(y,AF_TRELU,0.3);
   Print(y);

   Print("");
   vector a=vector::Full(x.Size(),2.0);
   x.Activation(y,AF_PRELU,a);
   Print(y);

/*  Results
   [0.1,0.4,0.9,2,-0.993262053000915,0,-0.095162581964040]
   [0.1,0.4,0.9,2,-1.986524106001829,0,-0.190325163928081]

   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.2,0.8,1.8,4,-10,0,-0.2]
   [5.2,5.8,6.8,9,-5,5,4.8]

   [0.1,0.4,0.9,2,-1.5,0,-0.03]
   [0.1,0.4,0.9,2,-5,0,-0.1]
   [0.1,0.4,0.9,2,-0.5,0,-0.01]

   [0.1,0.4,0.9,2,0,0,0]
   [0.2,0.8,0.9,2,-10,0,-0.2]
   [-1.8,-1.2,0.9,2,-12,-2,-2.2]

   [0.052497918747894,0.239475064044981,0.6398545523625035,1.761594155955765,-0.033464
   [0.054983399731247,0.275989792451045,0.7723340415895611,1.964027580075817,-0.000226

   [0,0,0,2,0,0,0]
   [0,0.4,0.9,2,0,0,0]

   [0.1,0.4,0.9,2,-10,0,-0.2]
*/
```

© 2000-2025, MetaQuotes Ltd.

Derivative

활성화 함수 도함수 값을 계산하고 이들을 전달된 벡터/행렬에 씁니다.

```
bool vector::Derivative(
  vector&                    vect_out,      // 값을 얻기 위한 벡터
  ENUM_ACTIVATION_FUNCTION   activation,    // 활성화 함수
   ...                                      // 추가 매개변수
   );


bool matrix::Derivative(
  matrix&                    matrix_out,    // 값을 얻기 위한 행렬
  ENUM_ACTIVATION_FUNCTION   activation,    // 활성화 함수
   );


bool matrix::Derivative(
  matrix&                    matrix_out,    // 값을 얻기 위한 행렬
  ENUM_ACTIVATION_FUNCTION   activation,    // 활성화 함수
  ENUM_MATRIX_AXIS           axis,          // 축
   ...                                      // 추가 매개변수
   );
```

매개 변수

vect_out/matrix_out

【out】활성화 함수의 도함수의 계산된 값을 얻기 위한 벡터 또는 행렬.

activation

[in] ENUM_ACTIVATION_FUNCTION 열거로 부터의 활성화 함수.

axis

[in]ENUM_MATRIX_AXIS 열거형 값(AXIS_HORZ — 가로축, AXIS_VERT — 세로축).

...

【in】추가 매개변수는 활성화 함수와의 추가 매개변수와 동일합니다. 일부의 활성화 함수만 추가 매개변수를 허용합니다. 매개변수를 지정하지 않으면 기본값이 사용됩니다.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환.

참조

함수 도함수는 오류 역전파 동안 학습에서 수신된 오류를 기반으로 모델 매개변수의 효율적인 업데이트를 가능하게 합니다.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Loss

손실 함수의 값을 계산합니다.

```
double vector::Loss(
  const vector&        vect_true,     // 참인 값의 벡터
  ENUM_LOSS_FUNCTION   loss,          // 손실 함수
   ...                                // 추가 매개변수
   );


double matrix::Loss(
  const matrix&        matrix_true,   // 참인 값의 행렬
  ENUM_LOSS_FUNCTION   loss,          // 손실 함수
   );


double matrix::Loss(
  const matrix&        matrix_true,   // 참인 값의 행렬
  ENUM_LOSS_FUNCTION   loss,          // 손실 함수
  ENUM_MATRIX_AXIS     axis,          // axis
   ...                                // 추가 매개변수
   );
```

매개 변수

vect_true/matrix_true

[in] 참인 값의 행렬 혹은 벡터.

loss

【n】ENUM_LOSS_FUNCTION 열거fh qnxjdml 손실 함수.

axis

[in]ENUM_MATRIX_AXIS 열거형 값(AXIS_HORZ —가로축, AXIS_VERT —세로축).

...

【n】 추가 매개변수 'delta'는 Hubert 손실 함수(LOSS_HUBER)에서만 사용할 수 있습니다.

Return Value

더블 값

Hubert 손실 함수(LOSS_HUBER)에서 'delta' 매개변수가 사용되는 방식

```
   double delta = 1.0;
   double error = fabs(y - x);
   if(error<delta)
      loss = 0.5 * error^2;
   else
      loss = 0.5 * delta^2 + delta * (error - delta);
```

© 2000-2025, MetaQuotes Ltd.

참조

신경망은 손실 함수가 사용되는 학습 샘플의 오류를 최소화하는 알고리즘을 찾는 것을 목표로 합니다.

손실 함수의 값은 모델이 예측한 값이 실제 값과 얼마나 다른지를 나타냅니다.

문제에 따라 각기 다른 손실 함수가 사용됩니다. 예를 들어 평균 제곱 오차(Mean Squared Error)(MSE)는 회귀 문제에 사용되며 Binary Cross-Entropy(BCE)는 이진 분류 목적으로 사용됩니다.

Hubert 손실 함수를 호출하는 예:

```
   vector y_true = {0.0, 1.0, 0.0, 0.0};
   vector y_pred = {0.6, 0.4, 0.4, 0.6};
   double loss=y_pred.Loss(y_true,LOSS_HUBER);
   Print(loss);
   double loss2=y_pred.Loss(y_true,LOSS_HUBER,0.5);
   Print(loss2);

/* Result
   0.155
   0.15125
*/
```

© 2000-2025, MetaQuotes Ltd.

LossGradient

손실 함수 기울기의 벡터 또는 행렬을 계산합니다.

```
vector vector::LossGradient(
  const vector&        vect_true,     // 참인 값의 벡터
  ENUM_LOSS_FUNCTION   loss,          // 손실 함수 종류
   ...                                // 추가 매개변수
   );


matrix matrix::LossGradient(
  const matrix&        matrix_true,   // 참인 값의 행렬
  ENUM_LOSS_FUNCTION   loss,          // 손실 함수
   );


matrix matrix::LossGradient(
  const matrix&        matrix_true,   // 참인 값의 행렬
  ENUM_LOSS_FUNCTION   loss,          // 손실 함수
  ENUM_MATRIX_AXIS     axis,          // axis
   ...                                // 추가 매개변수
   );
```

매개 변수

vect_true/matrix_true

[in] 참인 값의 행렬 혹은 벡터.

loss

【n】ENUM_LOSS_FUNCTION 열거로부터의 손실 함수.

axis

[in]ENUM_MATRIX_AXIS 열거형 값(AXIS_HORZ — 가로축, AXIS_VERT — 세로축).

...

【n】 추가 매개변수 'delta'는 Hubert 손실 함수(LOSS_HUBER)에서만 사용할 수 있습니다.

Return Value

손실 함수 기울기 값의 벡터 또는 행렬. 그래디언트는 주어진 지점에서 손실 함수의 dx(x는 예측값)에 대한 편도함수입니다.

참조

기울기는 신경망에서 모델을 훈련할 때 역전파 중에 가중치 행렬의 가중치를 조정하는 데 사용됩니다.

신경망은 손실 함수가 사용되는 학습 샘플의 오류를 최소화하는 알고리즘을 찾는 것을 목표로 합니다.

문제에 따라 각기 다른 손실 함수가 사용됩니다. 예를 들어 평균 제곱 오차(Mean Squared Error)(MSE)는 회귀 문제에 사용되며 Binary Cross-Entropy(BCE)는 이진 분류 목적으로 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

손실 함수 기울기 계산의 예

```
   matrixf y_true={{ 1, 2, 3, 4 },
                   { 5, 6, 7, 8 },
                   { 9,10,11,12 }};
   matrixf y_pred={{ 1, 2, 3, 4 },
                   {11,10, 9, 8 },
                   { 5, 6, 7,12 }};
   matrixf loss_gradient =y_pred.LossGradient(y_true,LOSS_MAE);
   matrixf loss_gradienth=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_HORZ);
   matrixf loss_gradientv=y_pred.LossGradient(y_true,LOSS_MAE,AXIS_VERT);
   Print("loss gradients\n",loss_gradient);
   Print("loss gradients on horizontal axis\n",loss_gradienth);
   Print("loss gradients on vertical axis\n",loss_gradientv);

/* Result
   loss gradients
   [[0,0,0,0]
    [0.083333336,0.083333336,0.083333336,0]
    [-0.083333336,-0.083333336,-0.083333336,0]]
   loss gradients on horizontal axis
   [[0,0,0,0]
    [0.33333334,0.33333334,0.33333334,0]
    [-0.33333334,-0.33333334,-0.33333334,0]]
   loss gradients on vertical axis
   [[0,0,0,0]
    [0.25,0.25,0.25,0]
    [-0.25,-0.25,-0.25,0]]
*/
```

© 2000-2025, MetaQuotes Ltd.

RegressionMetric

실제 데이터와 비교하여 예측 데이터의 품질을 평가하기 위해 회귀 메트릭을 계산합니다.

```
double vector::RegressionMetric(
   const vector&           vector_true,   // 참인 값의 벡터
   ENUM_REGRESSION_METRIC  metric         // 메트릭 타입
   );

double matrix::RegressionMetric(
   const matrix&           matrix_true,   // 참인 값의 행렬
   ENUM_REGRESSION_METRIC  metric         // 메트릭 타입
);

vector matrix::RegressionMetric(
   const matrix&           matrix_true,   // 참인 값의 행렬
   ENUM_REGRESSION_METRIC  metric,        // 메트릭 타입
   int                     axis           // 축
   );
```

패러미터

vector_true/matrix_true

【n】참 값의 벡터 또는 행렬입니다.

metric

【n】ENUM_REGRESSION_METRIC 열거의 메트릭 유형입니다.

axis

[in] 축. 0 — 가로축, 1 — 세로축.

반환 값

실제 데이터와 비교하여 예측 데이터의 품질을 평가하는 계산된 메트릭입니다.

주의

- REGRESSION_MAE — 예측 값과 해당 참 값 간의 절대 차이를 나타내는 평균 절대 오차(mean absolute error)
- REGRESSION_MSE — 예측 값과 해당 참 값 사이의 차이 제곱을 나타내는 평균 제곱 오차
- REGRESSION_RMSE — MSE의 제곱근
- REGRESSION_R2 - 1 – MSE(regression) / MSE(mean)
- REGRESSION_MAPE — 백분율인 MAE
- REGRESSION_MSPE — 백분율인 MSE
- REGRESSION_RMSLE — 로그 스케일로 계산된 RMSE

예:

```
   vector y_true = {3, -0.5, 2, 7};
```

© 2000-2025, MetaQuotes Ltd.

```
   vector y_pred = {2.5, 0.0, 2, 8};
//---
   double mse=y_pred.RegressionMetric(y_true,REGRESSION_MSE);
   Print("mse=",mse);
//---
   double mae=y_pred.RegressionMetric(y_true,REGRESSION_MAE);
   Print("mae=",mae);
//---
   double r2=y_pred.RegressionMetric(y_true,REGRESSION_R2);
   Print("r2=",r2);

  /* Result
   mae=0.375
   mse=0.5
   r2=0.9486081370449679
   */
```

© 2000-2025, MetaQuotes Ltd.

ConfusionMatrix

혼동 행렬 계산. 이 메서드는 예측값의 벡터에 적용됩니다.

```
행렬 벡터::ConfusionMatrix(
   const vector&        vect_true      // 참인 값의 벡터
   );


행렬 벡터::ConfusionMatrix(
   const vector&        vect_true,     // 참인 값들의 벡터
   uint                 label          // 레이블 값
   );
```

매개 변수

vect_true

[in] 참인 값들의 벡터.

label

[in] 혼동 행렬 계산을 위한 레이블 값.

반환값

혼동 행렬 레이블 값이 지정되지 않은 경우 다중 클래스 혼동 행렬이 반환되며 여기서 각 레이블은 개별적으로 서로 일치합니다. 레이블 값이 지정되면 지정된 레이블은 양수로 간주되고 다른 모든 레이블은 음수(ovr, 1 대 나머지)로 간주되는 2 x 2 행렬이 반환됩니다.

참조

혼동 행렬 C는 Cij가 그룹 i에 속하고 그룹 j에 속할 것으로 예측되는 관측치 수와 동일하도록 구성됩니다. 따라서 이진 분류에서 참음성(TN)의 개수는 C00, 위음성(FN)은 C10, 참양성(TP)은 C11, 위양성(FP)은 C01입니다.

즉, 행렬은 다음과 같이 그래픽으로 표현될 수 있습니다:

TN	FP
FN	TP

참값 벡터와 예측값 벡터의 크기는 동일해야 합니다.

예:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix confusion=y_pred.ConfusionMatrix(y_true);
   Print(confusion);
```

© 2000-2025, MetaQuotes Ltd.

```
   confusion=y_pred.ConfusionMatrix(y_true,0);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,1);
   Print(confusion);
   confusion=y_pred.ConfusionMatrix(y_true,2);
   Print(confusion);


/*
  [[3,0,0,0,0,0,0,0,0,0]
   [0,3,0,0,0,0,0,0,0,0]
   [0,0,1,0,1,0,0,1,0,0]
   [0,0,0,1,0,0,0,1,0,0]
   [0,0,1,0,3,0,0,0,0,1]
   [0,0,0,0,0,2,0,0,0,0]
   [1,0,0,0,0,1,1,0,0,0]
   [0,0,0,0,0,0,0,2,0,1]
   [0,0,1,0,0,0,0,0,0,1]
   [0,0,0,0,0,0,0,0,0,4]]
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
*/
```

© 2000-2025, MetaQuotes Ltd.

ConfusionMatrixMultilabel

각 레이블에 대한 혼동 행렬 계산. 이 메서드는 예측값의 벡터에 적용됩니다.

```
uint vector::ConfusionMatrixMultiLabel(
   const vector&        vect_true,     // 참인 값들의 벡터
   matrix&              confusions[]   // 계산된 혼동 행렬의 배열
   );
```

매개 변수

vect_true

[in] 참인 값들의 벡터.

confusions

[out] 각 레이블에 대해 계산된 혼동 행렬이 있는 2 x 2 행렬의 배열.

반환값

계산된 혼동 행렬 배열의 크기. 실패인 경우 0을 반환

참조

결과 배열은 동적이거나 정적일 수 있습니다. 배열이 정적이면 크기가 클래스의 수보다 작아서는 안 됩니다.

참값 벡터와 예측값 벡터의 크기는 동일해야 합니다.

예:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};
   matrix label_confusions[12];

   uint   res=y_pred.ConfusionMatrixMultiLabel(y_true,label_confusions);
   Print("res=",res,"  size=",label_confusions.Size());
   for(uint i=0; i<res; i++)
      Print(label_confusions[i]);


/*
  res=10  size=12
  [[26,1]
   [0,3]]
  [[27,0]
   [0,3]]
  [[25,2]
   [2,1]]
```

© 2000-2025, MetaQuotes Ltd.

```
   [[28,0]
    [1,1]]
   [[24,1]
    [2,3]]
   [[27,1]
    [0,2]]
   [[27,0]
    [2,1]]
   [[25,2]
    [1,2]]
   [[28,0]
    [2,0]]
   [[23,3]
    [0,4]]
*/
```

© 2000-2025, MetaQuotes Ltd.

ClassificationMetric

실제 데이터와 비교하여 예측 데이터의 품질을 평가하기 위해 분류 지표를 계산합니다. 이 방법은 예측 값의 벡터에 적용됩니다.

```
vector vector::ClassificationMetric(
   const vector&                vect_true,     // 참인 값들의 벡터
   ENUM_CLASSIFICATION_METRIC metric          // 메트릭 유형
   );


vector vector::ClassificationMetric(
   const vector&                vect_true,     // 참인 값들의 벡터
   ENUM_CLASSIFICATION_METRIC metric          // 메트릭 유형
   ENUM_AVERAGE_MODE          mode            // 평균화 모드
   );
```

매개 변수

vect_true

[in] 참인 값들의 벡터.

메트릭

[in] ENUM_CLASSIFICATION_METRIC 열거에서 메트릭 유형. CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION 및 CLASSIFICATION_ROC_AUC(ClassificationScore 메서드에 사용됨) 이외의 값이 적용됩니다.

모드

[in] ENUM_AVERAGE_MODE 열거의 평균화 모드. CLASSIFICATION_F1, CLASSIFICATION_JACCARD, CLASSIFICATION_PRECISION 및 CLASSIFICATION_RECALL 메트릭에 사용됩니다.

반환값

계산된 메트릭이 포함된 벡터. AVERAGE_NONE 평균화 모드의 경우 벡터에는 평균화 없이 각 클래스에 대한 메트릭 값이 포함됩니다. (예를 들어 이진 분류의 경우 이는 각각 'false'와 'true'에 대한 두 개의 메트릭이 됩니다).

평균화 모드에 대한 참고 사항

AVERAGE_BINARY는 이진 분류에만 의미가 있습니다.

AVERAGE_MICRO — 총 참양성, 거짓음성, 거짓양성을 계산하여 전역적으로 지표를 계산합니다.

AVERAGE_MACRO — 각 라벨의 메트릭을 계산하고 이들의 비가중 평균을 찾습니다. 이는 라벨 불균형을 고려하지 않습니다.

AVERAGE_WEIGHTED — 각 라벨의 메트릭을 계산하고 지원(각 라벨의 실제 인스턴스 수)을 기준으로 가중치가 부여된 평균을 찾습니다. 이는 라벨 불균형을 설명하기 위해 '매크로'를 변경합니다; 정밀도와 재현율 사이에 있지 않은 F 점수가 나올 수 있습니다.

참조

© 2000-2025, MetaQuotes Ltd.

이진 분류의 경우 첫 번째 열에 음수 레이블에 대한 확률이 포함되고 두 번째 열에 양수 레이블에 대한 확률이 포함된 n x 2 행렬 뿐만 아니라 양수 확률에 대한 하나의 열로 구성된 행렬도 입력할 수 있습니다. 이는 이진 분류 모델이 양성 라벨에 대해 두 개의 확률이나 하나의 확률을 반환할 수 있기 때문입니다..

예:

```
vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

vector accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
Print("accuracy=",accuracy);
vector balanced=y_pred.ClassificationMetric(y_true,CLASSIFICATION_BALANCED_ACCURACY
Print("balanced=",balanced);
Print("");

vector f1_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MICRO)
Print("f1_micro=",f1_micro);
vector f1_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_MACRO)
Print("f1_macro=",f1_macro);
vector f1_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_WEI
Print("f1_weighted=",f1_weighted);
vector f1_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_F1,AVERAGE_NONE);
Print("f1_none=",f1_none);
Print("");

vector jaccard_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVER
Print("jaccard_micro=",jaccard_micro);
vector jaccard_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVER
Print("jaccard_macro=",jaccard_macro);
vector jaccard_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,A
Print("jaccard_weighted=",jaccard_weighted);
vector jaccard_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_JACCARD,AVERA
Print("jaccard_none=",jaccard_none);
Print("");

vector precision_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_micro=",precision_micro);
vector precision_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,
Print("precision_macro=",precision_macro);
vector precision_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISI
Print("precision_weighted=",precision_weighted);
vector precision_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_PRECISION,A
Print("precision_none=",precision_none);
Print("");

vector recall_micro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
Print("recall_micro=",recall_micro);
```

© 2000-2025, MetaQuotes Ltd.

```
   vector recall_macro=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAG
   Print("recall_macro=",recall_macro);
   vector recall_weighted=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVE
   Print("recall_weighted=",recall_weighted);
   vector recall_none=y_pred.ClassificationMetric(y_true,CLASSIFICATION_RECALL,AVERAGE
   Print("recall_none=",recall_none);
   Print("");

//--- 바이너리 분류
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};

   vector f1_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_F1,AVERAGE_
   Print("f1_bin=",f1_bin);
   vector jaccard_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_JACCAR
   Print("jaccard_bin=",jaccard_bin);
   vector precision_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_PREC
   Print("precision_bin=",precision_bin);
   vector recall_bin=y_pred_bin.ClassificationMetric(y_true_bin,CLASSIFICATION_RECALL,
   Print("recall_bin=",recall_bin);


/*
  accuracy=[0.6666666666666666]
  balanced=[0.6433333333333333]

  f1_micro=[0.6666666666666666]
  f1_macro=[0.6122510822510823]
  f1_weighted=[0.632049062049062]
  f1_none=[0.8571428571428571,1,0.3333333333333333,0.6666666666666666,0.6666666666666

  jaccard_micro=[0.5]
  jaccard_macro=[0.4921428571428572]
  jaccard_weighted=[0.5056349206349205]
  jaccard_none=[0.75,1,0.2,0.5,0.5,0.6666666666666666,0.3333333333333333,0.4,0,0.5714

  precision_micro=[0.6666666666666666]
  precision_macro=[0.6571428571428571]
  precision_weighted=[0.6706349206349207]
  precision_none=[0.75,1,0.3333333333333333,1,0.75,0.6666666666666666,1,0.5,0,0.571428

  recall_micro=[0.6666666666666666]
  recall_macro=[0.6433333333333333]
  recall_weighted=[0.6666666666666666]
  recall_none=[1,1,0.3333333333333333,0.5,0.6,1,0.3333333333333333,0.6666666666666666,

  f1_bin=[0.4444444444444445]
  jaccard_bin=[0.2857142857142857]
  precision_bin=[0.5]
```

© 2000-2025, MetaQuotes Ltd.

```
   recall_bin=[0.4]
*/
```

© 2000-2025, MetaQuotes Ltd.

ClassificationScore

실제 데이터와 비교하여 예측 데이터의 품질을 평가하기 위해 분류 메트릭을 계산.

머신 러닝 섹션의 다른 메서드 달리 이 메서드는 예측 값 벡터가 아닌 실제 값 벡터에 적용됩니다.

```
vector vector::ClassificationScore(
   const matrix&              pred_scores,   // 각 클래스에 대한 확률 분포를 포함하는 행렬
   ENUM_CLASSIFICATION_METRIC metric         // 메트릭 유형
   ENUM_AVERAGE_MODE          mode           // 평균화 모드
   );


vector vector::ClassificationScore(
   const matrix&              pred_scores,   // 각 클래스에 대한 확률 분포를 포함하는 행렬
   ENUM_CLASSIFICATION_METRIC metric         // 메트릭 유형
   int                        param          // 추가 매개변수
   );
```

매개 변수

pred_scores

[in] 각 클래스에 대한 확률이 포함된 수평 벡터 세트를 포함하는 행렬. 행렬의 행의 수는 실제 값의 벡터 크기와 일치해야 합니다.

메트릭

[in] ENUM_CLASSIFICATION_METRIC 열거에서 메트릭 유형. CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION 및 CLASSIFICATION_ROC_AUC 값이 사용됩니다.

모드

[in] ENUM_AVERAGE_MODE 열거의 평균화 모드. CLASSIFICATION_AVERAGE_PRECISION 및 CLASSIFICATION_ROC_AUC 메트릭에 사용됩니다.

param

[in] CLASSIFICATION_TOP_K_ACCURACY 메트릭의 경우 평균화 모드 대신 정수 K 값을 지정해야 합니다.

반환값

계산된 메트릭이 포함된 벡터. AVERAGE_NONE 평균화 모드의 경우 벡터에는 평균화 없이 각 클래스에 대한 메트릭 값이 포함됩니다. (예를 들어 이진 분류의 경우 이는 각각 'false'와 'true'에 대한 두 개의 메트릭이 됩니다).

평균화 모드에 대한 참고 사항

AVERAGE_BINARY는 이진 분류에만 의미가 있습니다.

AVERAGE_MICRO — 레이블 지표 행렬의 각각의 요소를 레이블로 간주하여 전역적으로 측정항목을 계산합니다. 레이블 지표 행렬은 각 레이블에 대한 확률 집합이 포함된 행렬을 나타냅니다.

© 2000-2025, MetaQuotes Ltd.

AVERAGE_MACRO — 각 라벨의 메트릭을 계산하고 이들의 비가중 평균을 찾습니다. 이는 라벨 불균형을 고려하지 않습니다.

AVERAGE_WEIGHTED — 각 라벨의 메트릭을 계산하고 지원(각 라벨의 실제 인스턴스 수)을 기준으로 가중치가 부여된 평균을 찾습니다.

참조

이진 분류의 경우 첫 번째 열에 음수 레이블에 대한 확률이 포함되고 두 번째 열에 양수 레이블에 대한 확률이 포함된 n x 2 행렬 뿐만 아니라 양수 확률에 대한 하나의 열로 구성된 행렬도 입력할 수 있습니다. 이는 이진 분류 모델이 양성 라벨에 대해 두 개의 확률이나 하나의 확률을 반환할 수 있기 때문입니다..

예:

```
   vector y_true={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,8,4,2,7,6,8,4,2,3,6};
   //vector y_pred={7,2,1,0,4,1,4,9,5,9,0,6,9,0,1,5,9,7,3,4,2,9,4,9,5,9,2,7,7,0};

//--- 라벨 스코어          0         1         2         3         4         5
   matrix y_scores={{0.000109, 0.000186, 0.000449, 0.000052, 0.000002, 0.000022, 0.000
                    {0.000091, 0.081956, 0.916816, 0.001106, 0.000006, 0.000002, 0.000
                    {0.000108, 0.972863, 0.003600, 0.000021, 0.010479, 0.000015, 0.000
                    {0.925425, 0.000080, 0.002913, 0.000057, 0.000274, 0.000638, 0.063
                    {0.000060, 0.000126, 0.000006, 0.000000, 0.993513, 0.000000, 0.000
                    {0.000016, 0.982124, 0.000045, 0.000002, 0.008445, 0.000001, 0.000
                    {0.000000, 0.000040, 0.000001, 0.000000, 0.989395, 0.000167, 0.000
                    {0.000795, 0.002938, 0.023447, 0.007418, 0.021838, 0.002476, 0.000
                    {0.000091, 0.000226, 0.000038, 0.000007, 0.000048, 0.854910, 0.068
                    {0.000000, 0.000000, 0.000000, 0.000000, 0.003004, 0.000000, 0.000
                    {0.998856, 0.000009, 0.000976, 0.000002, 0.000000, 0.000013, 0.000
                    {0.000178, 0.000446, 0.000326, 0.000033, 0.000193, 0.000071, 0.998
                    {0.000005, 0.000016, 0.000153, 0.000045, 0.004110, 0.000012, 0.000
                    {0.994188, 0.000003, 0.002584, 0.000005, 0.000005, 0.000100, 0.000
                    {0.000173, 0.990569, 0.000792, 0.000040, 0.001798, 0.000035, 0.000
                    {0.000000, 0.000537, 0.000008, 0.005080, 0.000046, 0.992910, 0.000
                    {0.000127, 0.000003, 0.000003, 0.000000, 0.001583, 0.000000, 0.000
                    {0.000001, 0.000012, 0.000072, 0.000020, 0.000000, 0.000000, 0.000
                    {0.000020, 0.000105, 0.001139, 0.901343, 0.002132, 0.083873, 0.000
                    {0.000002, 0.000048, 0.000019, 0.000000, 0.999347, 0.000002, 0.000
                    {0.000059, 0.001344, 0.612502, 0.002749, 0.000229, 0.000678, 0.000
                    {0.000586, 0.000740, 0.001625, 0.000007, 0.269341, 0.000076, 0.016
                    {0.009547, 0.018055, 0.283795, 0.071079, 0.426074, 0.082335, 0.036
                    {0.002506, 0.002545, 0.001148, 0.005659, 0.020416, 0.000112, 0.006
                    {0.001263, 0.001769, 0.000293, 0.000011, 0.000302, 0.881768, 0.112
                    {0.002904, 0.002909, 0.013421, 0.001461, 0.007519, 0.001251, 0.000
                    {0.000055, 0.001080, 0.893158, 0.000000, 0.104492, 0.000159, 0.001
                    {0.000344, 0.002693, 0.071184, 0.000262, 0.000001, 0.000003, 0.000
                    {0.001404, 0.009375, 0.002638, 0.229189, 0.000064, 0.000896, 0.007
                    {0.491140, 0.000125, 0.000024, 0.000302, 0.000038, 0.034947, 0.473
```

© 2000-2025, MetaQuotes Ltd.

```
   vector top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,1);
   Print("top 1 accuracy score = ",top_k);
   top_k=y_true.ClassificationScore(y_scores,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top 2 accuracy score = ",top_k);
   vector y_true2={0, 1, 2, 2};
   matrix y_score2={{0.5, 0.2, 0.2},  // 0 is in top 2
                    {0.3, 0.4, 0.2},  // 1 is in top 2
                    {0.2, 0.4, 0.3},  // 2 is in top 2
                    {0.7, 0.2, 0.1}}; // 2 isn't in top 2
   top_k=y_true2.ClassificationScore(y_score2,CLASSIFICATION_TOP_K_ACCURACY,2);
   Print("top k = ",top_k);
   Print("");

   vector ap_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score micro = ",ap_micro);
   vector ap_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISIO
   Print("average precision score macro = ",ap_macro);
   vector ap_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score weighted = ",ap_weighted);
   vector ap_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_AVERAGE_PRECISION
   Print("average precision score none = ",ap_none);
   Print("");

   vector area_micro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score micro = ",area_micro);
   vector area_macro=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAG
   Print("roc auc score macro = ",area_macro);
   vector area_weighted=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score weighted = ",area_weighted);
   vector area_none=y_true.ClassificationScore(y_scores,CLASSIFICATION_ROC_AUC,AVERAGE
   Print("roc auc score none = ",area_none);
   Print("");

//--- 바이너리 분류
   vector y_pred_bin={0,1,0,1,1,0,0,0,1};
   vector y_true_bin={1,0,0,0,1,0,1,1,1};
   vector y_score_true={0.3,0.7,0.1,0.6,0.9,0.0,0.4,0.2,0.8};
   matrix y_score1_bin(y_score_true.Size(),1);
   y_score1_bin.Col(y_score_true,0);
   matrix y_scores_bin={{0.7, 0.3},
                        {0.3, 0.7},
                        {0.9, 0.1},
                        {0.4, 0.6},
                        {0.1, 0.9},
                        {1.0, 0.0},
                        {0.6, 0.4},
                        {0.8, 0.2},
                        {0.2, 0.8}};
```

© 2000-2025, MetaQuotes Ltd.

```
   vector ap=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECIS
   Print("average precision score binary = ",ap);
   vector ap2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_AVERAGE_PREC
   Print("average precision score binary = ",ap2);
   vector ap3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PREC
   Print("average precision score none = ",ap3);
   Print("");

   vector area=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score binary = ",area);
   vector area2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_ROC_AUC,AV
   Print("roc auc score binary = ",area2);
   vector area3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AV
   Print("roc auc score none = ",area3);


/*
  top 1 accuracy score = [0.6666666666666666]
  top 2 accuracy score = [1]
  top k = [0.75]

  average precision score micro = [0.8513333333333333]
  average precision score macro = [0.9326666666666666]
  average precision score weighted = [0.9333333333333333]
  average precision score none = [1,1,0.7,1,0.9266666666666666,0.8333333333333333,1,0.

  roc auc score micro = [0.9839506172839506]
  roc auc score macro = [0.9892068783068803]
  roc auc score weighted = [0.9887354497354497]
  roc auc score none = [1,1,0.9506172839506173,1,0.984,0.9821428571428571,1,0.9753086

  average precision score binary = [0.7961904761904761]
  average precision score binary = [0.7961904761904761]
  average precision score none = [0.7678571428571428,0.7961904761904761]

  roc auc score binary = [0.7]
  roc auc score binary = [0.7]
  roc auc score none = [0.7,0.7]
*/
```

© 2000-2025, MetaQuotes Ltd.

PrecisionRecall

값들을 계산하여 정밀도-재현율 곡선(precision-recall curve)을 구성합니다. ClassificationScore와 마찬가지로 이 메서드는 참값의 벡터에 적용됩니다.

```
bool vector::PrecisionRecall(
   const matrix&                 pred_scores,   // 각 클래스의 확률 분포를 포함하는 행렬
   const ENUM_ENUM_AVERAGE_MODE  mode           // 평균화 모드
   matrix&                       precision,     // // 각 임계값에 대해 계산된 정밀도의 값들
   matrix&                       recall,        // 각 임계값에 대해 계산된 재현율 값
   matrix&                       thresholds,    // 내림차순으로 정렬된 기준값들
   );
```

매개 변수

pred_scores

[in] 각 클래스에 대한 확률이 포함된 수평 벡터의 집합을 포함하는 행렬. 행렬 행의 수는 실제 값으로 구성된 벡터의 크기와 일치해야 합니다.

mode

[in] ENUM_AVERAGE_MODE 열거의 평균화 모드. AVERAGE_NONE, AVERAGE_BINARY 및 AVERAGE_MICRO만 사용됩니다.

precision

[out] 계산된 정밀도 곡선 값이 포함된 행렬입니다. 평균화가 적용되지 않은 경우(AVERAGE_NONE) 행렬의 행 수는 모델의 클래스의 수에 해당합니다. 열 수는 참값 벡터의 크기(또는 확률 분포 행렬 pred_score의 행 수)에 해당합니다. 미세 평균화의 경우 행렬의 행 수는 중복을 제외한 임계값의 총 수에 해당합니다.

recall

[out]계산된 재현율 곡선 값이 포함된 행렬 .

threshold

[out] 확률 행렬을 정렬하여 얻은 임계 행렬

참조

ClassificationScore 메서드에 대한 참고 사항을 참조하세요.

예시

mnist.onnx 모델에서 통계를 수집하는 예(정확도 99%)

```
//--- 분류 측정항목 데이터
   vectorf y_true(images);
   vectorf y_pred(images);
   matrixf y_scores(images,10);
//--- 입력-출력
   matrixf image(28,28);
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf result(10);

//--- 테스팅
   for(int test=0; test<images; test++)
     {
      image=test_data[test].image;
      if(!OnnxRun(model,ONNX_DEFAULT,image,result))
        {
         Print("OnnxRun error ",GetLastError());
         break;
        }
      result.Activation(result,AF_SOFTMAX);
     //--- 데이터 수집
      y_true[test]=(float)test_data[test].label;
      y_pred[test]=(float)result.ArgMax();
      y_scores.Row(result,test);
     }    }
```

정확도 계산

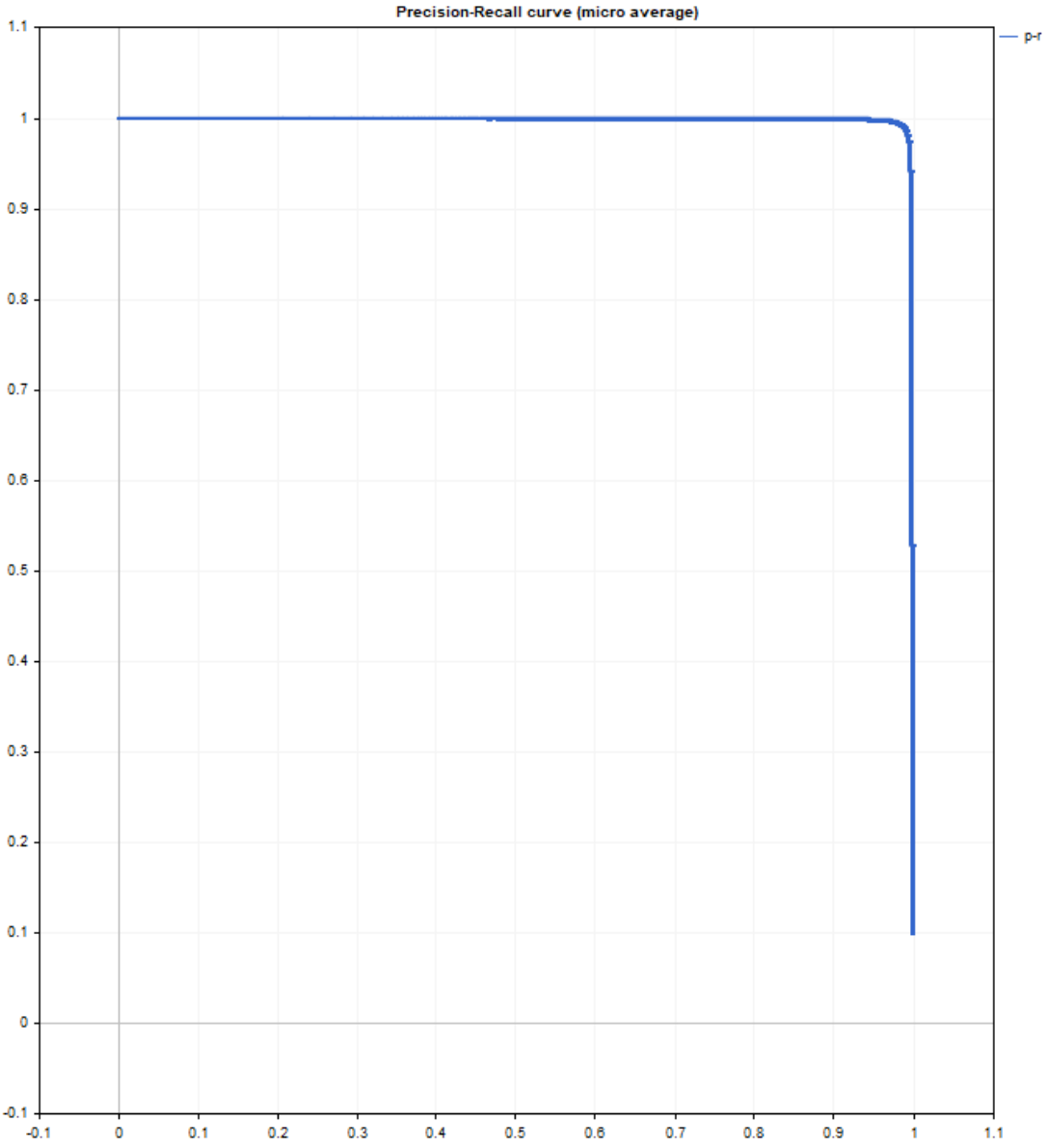
```
   vectorf accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   PrintFormat("accuracy=%f",accuracy[0]);

accuracy=0.989000
```

정밀도 값이 y축에 표시되고 재현율 값이 x축에 표시되는 정밀도-재현율 그래프를 그리는 예입니다. 또한 정밀도 및 재현율 그래프는 x축에 임계값이 표시되어 별도로 표시됩니다.

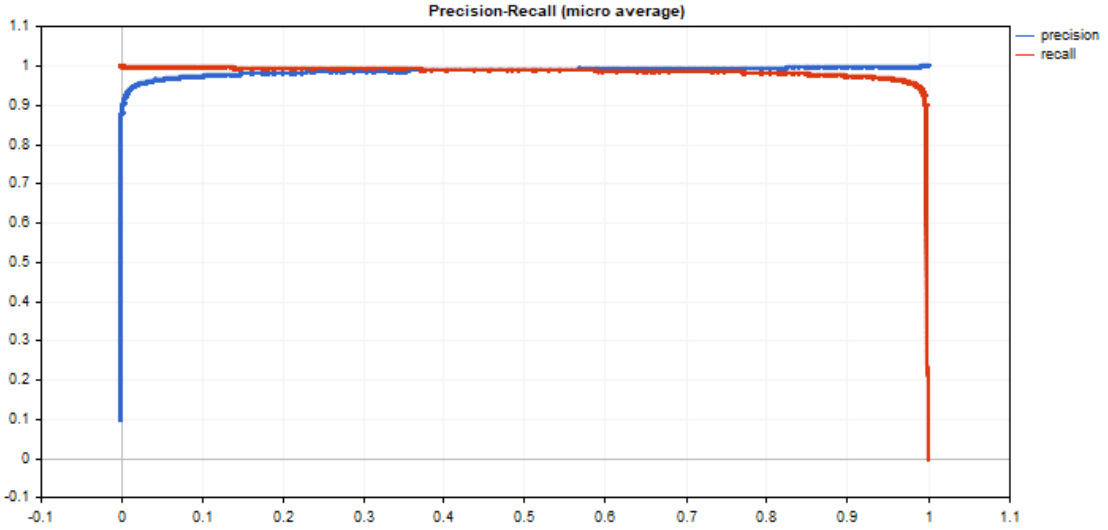
```
   if(y_true.PrecisionRecall(y_scores,AVERAGE_MICRO,mat_precision,mat_recall,mat_thres
     {
      double precision[],recall[],thres[];
      ArrayResize(precision,mat_thres.Cols());
      ArrayResize(recall,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<thres.Size(); i++)
        {
         precision[i]=mat_precision[0][i];
         recall[i]=mat_recall[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("Precision-Recall curve (micro average)","p-r","",recall,precision);
      Plot2Curves("Precision-Recall (micro average)","precision","recall",thres,precis
     }
```

© 2000-2025, MetaQuotes Ltd.

결과 곡선:



© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

ReceiverOperatingCharacteristic

Receiver Operating Characteristic (ROC) 곡선을 구성하기 위한 값을 계산합니다. ClassificationScore와 마찬가지로 이 메서드는 참값의 벡터에 적용됩니다.

```
bool vector::ReceiverOperatingCharacteristic(
   const 행렬&                  pred_scores,   // 각각의 클래스의 확률 분포를 포함하는 행렬
   const ENUM_ENUM_AVERAGE_MODE  mode           // 평균화 모드
   행렬&                        fpr,           // 각 임계값에 대해 계산된 거짓양성률의 값들
   행렬&                        tpr,           // 각 임계값에 대해 계산된 참양성 비율의 값들
   matrix&                      thresholds,    // 내림차순으로 정렬된 기준값들
   );
```

매개 변수

pred_scores

[in] 각 클래스에 대한 확률이 포함된 수평 벡터의 집합을 포함하는 행렬. 행렬 행의 수는 실제 값으로 구성된 벡터의 크기와 일치해야 합니다.

mode

[in] ENUM_AVERAGE_MODE 열거의 평균화 모드. AVERAGE_NONE, AVERAGE_BINARY 및 AVERAGE_MICRO만 사용됩니다.

fpr

[out] 거짓 양성률 곡선의 계산된 값이 포함된 행렬입니다. 평균화가 적용되지 않은 경우 (AVERAGE_NONE) 행렬의 행 수는 모델의 클래스의 수에 해당합니다. 열 수는 참값 벡터의 크기(또는 확률 분포 행렬 pred_score의 행 수)에 해당합니다. 미세 평균화의 경우 행렬의 행 수는 중복을 제외한 임계값의 총 수에 해당합니다.

tpr

[out] 진양성 비율 곡선의 계산된 값이 포함된 행렬입니다.

threshold

[out] 확률 행렬을 정렬하여 얻은 임계 행렬

참조

ClassificationScore 메서드에 대한 참고 사항을 참조하세요.

예시

ROC 그래프를 그리는 예. tpr 값은 y축에 표시되고 fpr 값은 x축에 표시됩니다. 또한 fpr 및 tpr 그래프는 x축에 임계값이 표시되며 별도로 표시됩니다.

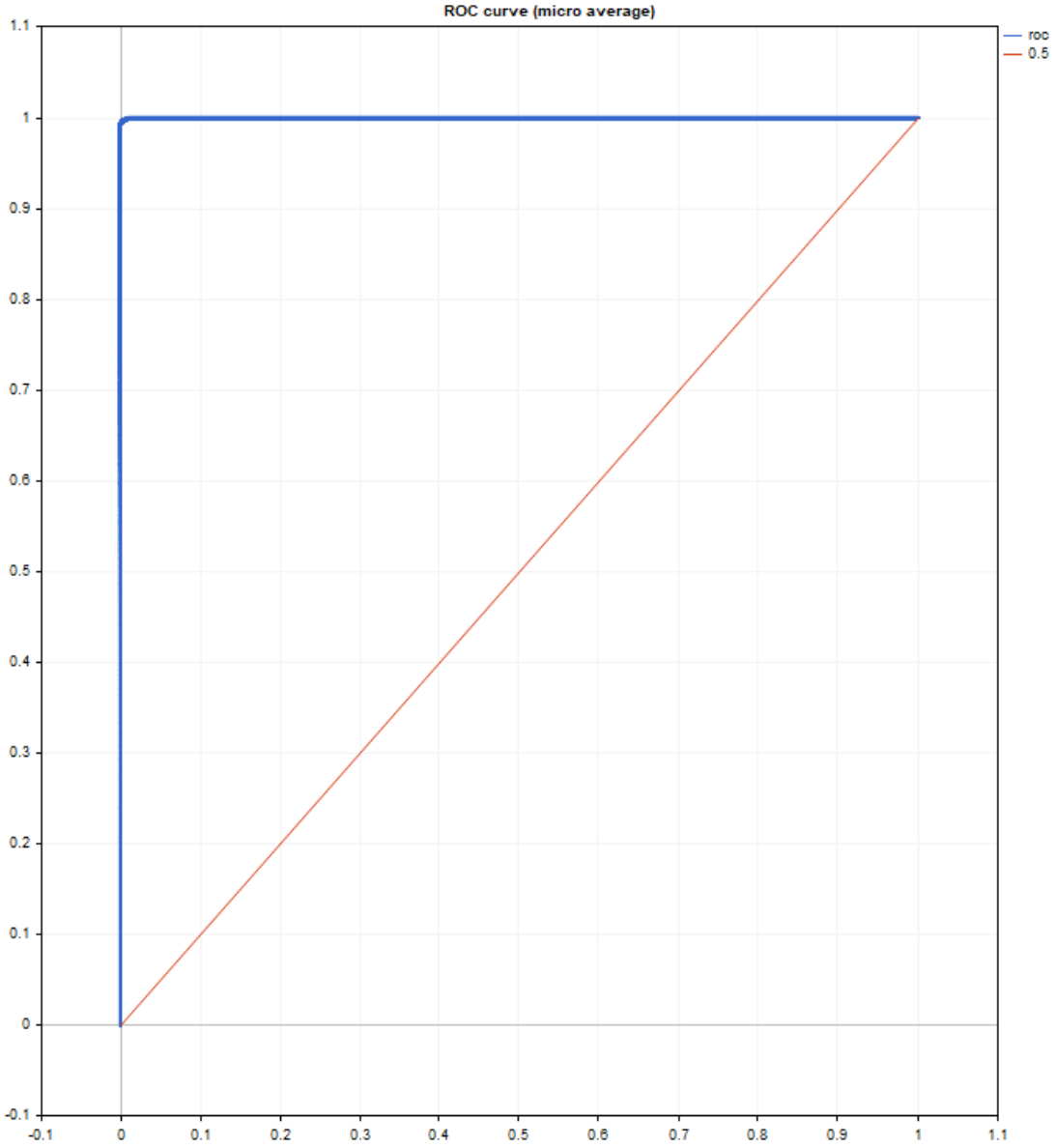
```
   matrixf mat_thres;
   matrixf mat_fpr;
   matrixf mat_tpr;

   if(y_true.ReceiverOperatingCharacteristic(y_scores,AVERAGE_MICRO,mat_fpr,mat_tpr,ma
     {
```

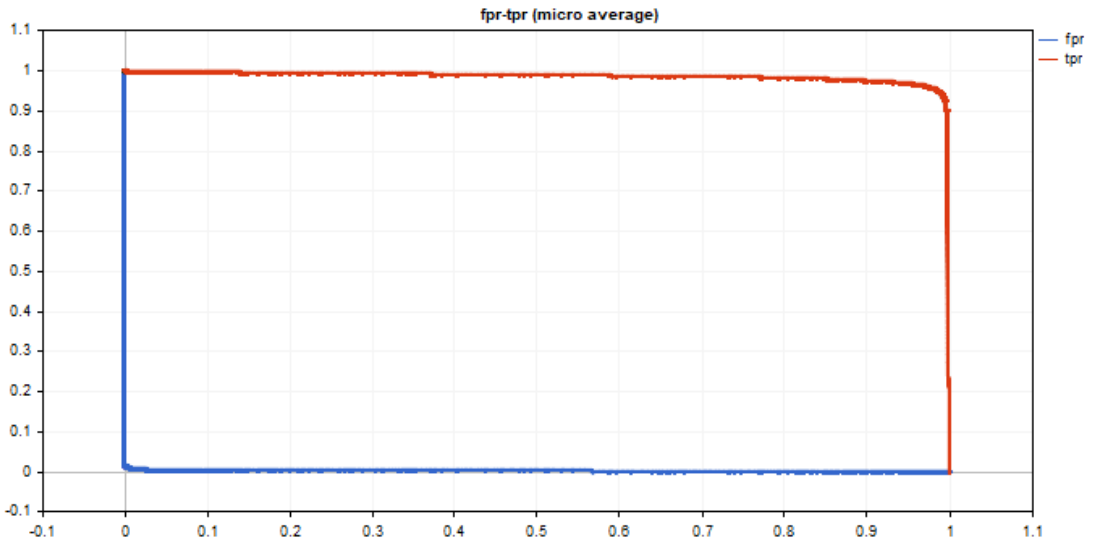
© 2000-2025, MetaQuotes Ltd.

```
      double fpr[],tpr[],thres[];
      ArrayResize(fpr,mat_thres.Cols());
      ArrayResize(tpr,mat_thres.Cols());
      ArrayResize(thres,mat_thres.Cols());

      for(uint i=0; i<fpr.Size(); i++)
        {
         fpr[i]=mat_fpr[0][i];
         tpr[i]=mat_tpr[0][i];
         thres[i]=mat_thres[0][i];
        }
      thres[0]=thres[1]+0.001;

      PlotCurve("ROC curve (micro average)","roc","0.5",fpr,tpr);
      Plot2Curves("fpr-tpr (micro average)","fpr","tpr",thres,fpr,tpr);
     }
```

결과 곡선:

© 2000-2025, MetaQuotes Ltd.





© 2000-2025, MetaQuotes Ltd.

그래프 출력 코드는 간단하며 <Graphics/Graphic.mqh> 표준 라이브러리를 기반으로 합니다.

예제는 mnist.onnx 모델의 데이터를 사용합니다. 코드는 PrecisionRecall 메서드 설명에 나와 있습니다.

ROC AUC가 이상적입니다.

```
roc auc score micro = [0.99991]
```

© 2000-2025, MetaQuotes Ltd.

OpenBLAS Methods

OpenBLAS is a high-performance open-source linear algebra library that implements BLAS (Basic Linear Algebra Subprograms) and some LAPACK functions. OpenBLAS is designed to improve computational performance, particularly in matrix and vector operations, which are often used in scientific and engineering tasks such as machine learning, numerical methods, and simulations.

Key features of OpenBLAS:

- Multithreading support: OpenBLAS can efficiently use multiple processor cores for parallel computations, significantly accelerating operations on multiprocessor systems.
- Optimization for processor architectures: OpenBLAS includes optimized builds for various processors such as Intel, AMD, ARM and others. The library automatically detects processor characteristics and selects the most suitable function implementations.
- Extensive BLAS operation support: OpenBLAS implements core BLAS functions, including vector operations (e.g., vector addition and dot product), matrix operations (multiplication), and vector-matrix operations.
- LAPACK compatibility: The library supports LAPACK (Linear Algebra PACKage) functions for more complex linear algebra operations, such as solving systems of linear equations, calculating matrix eigenvalues, and others.
- High performance: Compared to other BLAS libraries, OpenBLAS often shows better results due to hand-optimization for specific processor architectures.

Applications

OpenBLAS is widely used in applications involving numerical computations:

- Training neural networks and other machine learning tasks.
- Scientific computing (e.g. modeling of physical processes).
- Processing and analyzing large amounts of data.

The library is integrated into many popular scientific software packages such as NumPy, SciPy, and TensorFlow, which rely on high-performance linear algebra operations.

OpenBLAS is an excellent choice for those seeking an open-source solution for high-performance computing, particularly when working with large matrices and vectors.

Function	Action
SingularValueDecompositionDC	Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (LAPACK function GESDD).
SingularValueDecompositionQR	Singular Value Decomposition, QR algorithm. This algorithm is considered a classical SVD algorithm (LAPACK function GESVD).

© 2000–2025, MetaQuotes Ltd.

Function	Action
SingularValueDecompositionQRPivot	Singular Value Decomposition, QR with pivoting algorithm (LAPACK function GESVDQ).
SingularValueDecompositionBisect	Singular Value Decomposition, bisection algorithm (LAPACK function GESVDX).
SingularValueDecompositionJacobiHigh	Singular Value Decomposition, Jacobi high level algorithm (LAPACK function GEJSV).
SingularValueDecompositionJacobiLow	Singular Value Decomposition, Jacobi low level algorithm (LAPACK function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.
SingularValueDecompositionBidiagDC	Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (LAPACK function BDSVDX).
SingularValueDecompositionBidiagBisect	Singular Value Decomposition, bisection algorithm for bidiagonal matrices (LAPACK function BDSVDX).
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (LAPACK function GEEV).
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (LAPACK function GGEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (LAPACK function GEEVX).
EigenSolverSchur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (LAPACK function GEES). See also Schur decomposition.
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (LAPACK functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex

© 2000–2025, MetaQuotes Ltd.

Function	Action
	conjugate) matrix using the QR algorithm (LAPACK functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (LAPACK functions SYEVX, HEEVX).
SingularSpectrumAnalysisSpectrum	A method function for calculating the relative contributions of spectral components based on their eigenvalues.
SingularSpectrumAnalysisForecast	A method function for calculating reconstructed and predicted data using spectral components of the input time series.
SingularSpectrumAnalysisReconstructComponents	A method function for calculating reconstructed components of the input time series and their contributions.
SingularSpectrumAnalysisReconstructSeries	A method function for calculating the reconstructed time series using the first component_count components.

© 2000–2025, MetaQuotes Ltd.

Singular value decomposition

This section features functions for decomposing a matrix into three components: orthogonal matrices and a diagonal matrix of singular values. SVD is applied to solve various linear algebra problems such as data dimensionality reduction, image compression, solving systems of equations, and data analysis and optimization. The main functions allow you to compute singular values and vectors, reconstruct matrices, and approximate matrices with reduced rank accuracy.

Function	Action
SingularValueDecompositionDC	Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (LAPACK function GESDD).
SingularValueDecompositionQR	Singular Value Decomposition, QR algorithm. This algorithm is considered a classical SVD algorithm (LAPACK function GESVD).
SingularValueDecompositionQRPivot	Singular Value Decomposition, QR with pivoting algorithm (LAPACK function GESVDQ).
SingularValueDecompositionBisect	Singular Value Decomposition, bisection algorithm (LAPACK function GESVDX).
SingularValueDecompositionJacobiHigh	Singular Value Decomposition, Jacobi high level algorithm (LAPACK function GEJSV).
SingularValueDecompositionJacobiLow	Singular Value Decomposition, Jacobi low level algorithm (LAPACK function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.
SingularValueDecompositionBidiagDC	Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (LAPACK function BDSDC).
SingularValueDecompositionBidiagBisect	Singular Value Decomposition, bisection algorithm for bidiagonal matrices (LAPACK function BDSVDX).
SingularValueDecompositionBidiagQR	Computes the singular value decomposition of a general matrix that has been reduced to bidiagonal form by the method ReduceToBidiagonal. LAPACK function BDSQR.

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionDC

Singular Value Decomposition, "divide-and-conquer" algorithm. This algorithm is considered the fastest among other SVD algorithms (LAPACK function GESDD).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z       jobz,      // how to computed
   vector&          S,         // vector of computed singular values
   matrix&          U,         // matrix of computed left vectors U
   matrix&          VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z       jobz,      // how to computed
   vectorf&         S,         // vector of computed singular values
   matrixf&         U,         // matrix of computed left vectors U
   matrixf&         VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z       jobz,      // how to computed
   vector&          S,         // vector of computed singular values
   matrixc&         U,         // matrix of computed left vectors U
   matrixc&         VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionDC(
   ENUM_SVD_Z       jobz,                  // how to computed
   vectorf&         singular_values,       // vector of computed singular values
   matrixcf&        U,                     // matrix of computed left vectors U
   matrixcf&        VT                     // transposed matrix of right vectors VT
   );
```

Parameters

jobz

[in] ENUM_SVD_Z enumeration value which determines the method for computing left and singular eigenvectors.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

© 2000–2025, MetaQuotes Ltd.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobz parameter.

When jobv is set to SVDZ_N, the left and right vectors are not computed. Only singular values are computed.

When jobv is set to SVDZ_A, the full matrices of the U and VT vectors are computed.

When the value is SVDZ_S, truncated matrices of vectors U and VT are computed.

ENUM_SVD_Z

An enumeration defining the way to compute left and right singular vectors.

ID	Description
SVDZ_N	Columns U or rows VT are not computed
SVDZ_A	All M columns of U or all N columns of VT are returned in arrays U and VT
SVDZ_S	The first min(M,N) columns of U or the first min(M,N) columns of VT are returned in arrays U and VT

See also

SingularValueDecompositionQR, SingularValueDecompositionQRPivot

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionQR

Singular Value Decomposition, QR algorithm. Considered a classical SVD algorithm (LAPACK function GESVD).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // how to compute left vectors
   ENUM_SVD_Z      jobv,     // how to compute right vectors
   vector&         S,        // vector of computed singular values
   matrix&         U,        // matrix of computed left vectors U
   matrix&         VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // how to compute left vectors
   ENUM_SVD_Z      jobv,     // how to compute right vectors
   vectorf&        S,        // vector of computed singular values
   matrixf&        U,        // matrix of computed left vectors U
   matrixf&        VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,     // how to compute left vectors
   ENUM_SVD_Z      jobv,     // how to compute right vectors
   vector          S,        // vector of computed singular values
   matrixc         U,        // matrix of computed left vectors U
   matrixc         VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionQR(
   ENUM_SVD_Z      jobu,                   // how to compute left vectors
   ENUM_SVD_Z      jobv,                   // how to compute right vectors
   vectorf&        singular_values,        // vector of computed singular values
   matrixcf&       u,                      // matrix of computed left vectors U
   matrixcf&       vt                      // transposed matrix of right vectors VT
   );
```

Parameters

jobu

[in] ENUM_SVD_Z enumeration value defining how the left singular vectors should be computed.

jobv

© 2000–2025, MetaQuotes Ltd.

[in] ENUM_SVD_Z enumeration value defining how the right singular vectors should be computed.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobu and jobv parameters.

When set to SVDZ_N, the left (jobu) and right (jobv) vectors are not computed. Singular values are always computed.

When set to SVDZ_A, the full matrices of vectors U (jobu) or VT (jobv) are computed.

With the value SVDZ_S, truncated matrices of vectors U (jobu) or VT (jobv) are computed.

ENUM_SVD_Z

An enumeration defining the way to compute left and right singular vectors.

ID	Description
SVDZ_N	Columns U or rows VT are not computed
SVDZ_A	All M columns of U or all N columns of VT are returned in arrays U and VT
SVDZ_S	The first min(M,N) columns of U or the first min(M,N) columns of VT are returned in arrays U and VT

See also

SingularValueDecompositionDC, SingularValueDecompositionQRPivot

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionQRPivot

Singular Value Decomposition, QR with pivoting algorithm (LAPACK function GESVDQ).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vector&         S,        // vector of computed singular values
   matrix&         U,        // matrix of computed left vectors U
   matrix&         VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vectorf&        S,        // vector of computed singular values
   matrixf&        U,        // matrix of computed left vectors U
   matrixf&        VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,     // computation accuracy level
   ENUM_SVDQRP_P   jobp,     // use row reversal to compute
   ENUM_SVDQRP_R   jobr,     // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,     // how to compute left vectors
   ENUM_SVDQRP_V   jobv,     // how to compute right vectors
   vector&         S,        // vector of computed singular values
   matrixc&        U,        // matrix of computed left vectors U
   matrixc&        VT        // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionQRPivot(
   ENUM_SVDQRP_A   joba,                  // computation accuracy level
   ENUM_SVDQRP_P   jobp,                  // use row reversal to compute
   ENUM_SVDQRP_R   jobr,                  // use triangular matrix R to compute
   ENUM_SVDQRP_U   jobu,                  // how to compute left vectors
```

© 2000–2025, MetaQuotes Ltd.

```
   ENUM_SVDQRP_V   jobv,                  // how to compute right vectors
   vectorf&        singular_values,       // vector of computed singular values
   matrixcf&       u,                     // matrix of computed left vectors U
   matrixcf&       vt                     // transposed matrix of right vectors VT
   );
```

Parameters

joba

[in] ENUM_SVDQRP_A enumeration value defining the accuracy level of the SVD computation.

jobp

[in] ENUM_SVDQRP_P enumeration value defining the use of row reversal during the computation process.

jobr

[in] ENUM_SVDQRP_R enumeration value defining whether to transpose the triangular matrix R obtained as a result of the initial QR factorization.

jobu

[in] ENUM_SVDQRP_U enumeration value that determines how the left singular vectors should be computed.

jobv

[in] ENUM_SVDQRP_V enumeration value defining how the right singular vectors should be computed.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

If both left and right singular vectors are computed, jobv must be set to SVDQRPV_R when jobu=SVDQRPU_R.

ENUM_SVDQRP_A

An enumeration that specifies the level of accuracy of the SVD computation.

© 2000–2025, MetaQuotes Ltd.

ID	Description
SVDQRP_A	The requested accuracy corresponds to the inverse error limited by epsilon. This is an aggressive level of truncation.
SVDQRPA_M	Similar to SVDQRP_A, but the truncation is softer. This is the average level of truncation.
SVDQRPA_H	High accuracy is required.

ENUM_SVDQRP_P

An enumeration that specifies whether to use row reversal during calculation.

ID	Description
SVDQRPP_P	The rows of A are ordered in descending order. Recommended for numerical reliability.
SVDQRPP_N	No row reversal.

ENUM_SVDQRP_R

An enumeration that specifies whether to transpose the triangular matrix R obtained as a result of the initial QR factorization.

ID	Description
SVDQRPR_T	After the initial rotated QR factorization, GESVD is applied to the transpose R**T of the calculated triangular factor R.
SVDQRPR_N	The triangular factor R is given as input to GESVD.

ENUM_SVDQRP_U

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDQRPU_A	All M left singular vectors are computed.
SVDQRPU_S	The min(M,N) left singular vectors are computed.
SVDQRPU_R	The numerical rank NUMRANK is determined and only the NUMRANK of the left singular vectors is computed.
SVDQRPU_F	N left singular vectors are returned.
SVDQRPU_N	Left singular vectors are not computed.

ENUM_SVDQRP_V

An enumeration defining how right singular vectors should be computed.

© 2000-2025, MetaQuotes Ltd.

ID	Description
SVDQRPV_A	All N right singular vectors are computed.
SVDQRPV_R	The numerical rank NUMRANK is defined and only the NUMRANK of right singular vectors are computed.
SVDQRPV_N	Right singular vectors are not computed.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionBisect

Singular Value Decomposition, bisection algorithm (LAPACK function GESVDX).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // computation accuracy level
   ENUM_BLAS_RANGE      range,     // subset of computable singular values
   double               lower,     // lower limit of the subset
   double               upper,     // upper limit of the subset
   vector&              S,         // vector of computed singular values
   matrix&              U,         // matrix of computed left vectors U
   matrix&              VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // computation accuracy level
   ENUM_BLAS_RANGE      range,     // subset of computable singular values
   double               lower,     // lower limit of the subset
   double               upper,     // upper limit of the subset
   vectorf&             S,         // vector of computed singular values
   matrixf&             U,         // matrix of computed left vectors U
   matrixf&             VT         // ransposed matrix of right vectors VT
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,      // computation accuracy level
   ENUM_BLAS_RANGE      range,     // subset of computable singular values
   double               lower,     // lower limit of the subset
   double               upper,     // upper limit of the subset
   vector               S,         // vector of computed singular values
   matrixc              U,         // matrix of computed left vectors U
   matrixc              VT         // transposed matrix of right vectors VT
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionBisect(
   ENUM_SVD_VECTORS     jobv,               // computation accuracy level
   ENUM_BLAS_RANGE      range,              // subset of computable singular values
   double               lower,              // lower limit of the subset
   double               upper,              // upper limit of the subset
   vectorf&             singular_values,    // vector of computed singular values
   matrixcf&            u,                  // matrix of computed left vectors U
   matrixcf&            vt                  // transposed matrix of right vectors VT
```

© 2000–2025, MetaQuotes Ltd.

```
   );
```

Parameters

jobv

[in] ENUM_SVD_VECTORS enumeration value which determines the method for computing left and singular eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobuv* and *range* parameters.

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

ENUM_SVD_VECTORS

An enumeration defining the way to compute left and right singular vectors.

© 2000–2025, MetaQuotes Ltd.

ID	Description
SVDVECTORS_N	Only singular values are computed, without vectors.
SVDVECTORS_U	Left singular vectors are computed.
SVDVECTORS_V	Right singular vectors are computed.
SVDVECTORS_UV	Left and right singular vectors are computed.

ENUM_BLAS_RANGE

An enumeration defining a subset of computable singular values and vectors.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	Singular or eigenvalues from IL to IU will be found

See also

SingularValueDecompositionDC, SingularValueDecompositionQR, SingularValueDecompositionQRPivot

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionJacobiHigh

Singular Value Decomposition, Jacobi high level algorithm (LAPACK function GEJSV).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a sq
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to
   vector&        S,               // vector of computed singular values
   matrix&        U,               // matrix of computed left vectors U
   matrix&        V,               // matrix of computed left vectors V
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a sq
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to
   vectorf&       S,               // vector of computed singular values
   matrixf&       U,               // U matrix of computed left vectors
   matrixf&       V,               // V matrix of computed left vectors
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A   joba,            // computation accuracy level
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   ENUM_SVDJH_R   jobr,            // define range of computable singular values
   ENUM_SVDJH_T   jobt,            // define whether to transpose when computing a sq
   ENUM_SVDJH_P   jobp,            // the possibility of structured perturbations to
   vectorc&       S,               // vector of computed singular values
   matrixc&       U,               // U matrix of computed left vectors
   matrixc&       V,               // V matrix of computed left vectors
   vector&        work_results     // additional computation results
   );
```

© 2000–2025, MetaQuotes Ltd.

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionJacobiHigh(
   ENUM_SVDJH_A    joba,                  // computation accuracy level
   ENUM_SVDJH_U    jobu,                  // how to compute left vectors
   ENUM_SVDJH_V    jobv,                  // how to compute right vectors
   ENUM_SVDJH_R    jobr,                  // define range of computable singular valu
   ENUM_SVDJH_T    jobt,                  // define whether to transpose when computi
   ENUM_SVDJH_P    jobp,                  // the possibility of structured perturbati
   vectorf&        singular_values,       // vector of computed singular values S
   matrixcf&       u,                     // matrix of computed left vectors U
   matrixcf&       v,                     // matrix of computed left vectors V
   vectorf&        work_results           // additional computation results
   );
```

Parameters

joba

[in] ENUM_SVDJH_A enumeration value determining the accuracy level of the SVD computation.

jobu

[in] ENUM_SVDJH_U enumeration value that determines how the left singular vectors should be computed.

jobv

[in] ENUM_SVDJH_V enumeration value that determines how the right singular vectors should be computed.

jobr

[in] ENUM_SVDJH_R enumeration value defining the range of computable values

jobt

[in] ENUM_SVDJH_T enumeration value defining whether to transpose the matrix of it is square. If the matrix is non-square, this parameter is ignored.

jobp

[in] ENUM_SVDJH_P enumeration value defining the possibility of structured perturbations to remove denormalized values.

S

[out] Vector of singular values.

For work(1)/work(2) = one: the singular values of A. During the computation S contains Euclidean column norms of the iterated matrices in the array a.

For work(1)≠work(2): the singular values of A are (work(1)/work(2)) * S(1:n). This factored form is used if sigma_max(A) overflows or if small singular values have been saved from underflow by scaling the input matrix A.

jobr = 'R', some of the singular values may be returned as exact zeros obtained by 'setting to zero' because they are below the numerical rank threshold or are denormalized numbers.

U

© 2000–2025, MetaQuotes Ltd.

[out] Matrix of left singular vectors.

V

[out] Matrix of right singular vectors.

work_results

[out] Vector consisting of 7 statistics obtained as a result of the computation.

work(1) = scale = work(2)/work(1) is the scaling factor such that scale*sva(1:n) are the computed singular values of A. See the description of S.

work(2) = see the description of work(1).

work(3) = sconda is an estimate for the condition number of column equilibrated A. If joba = 'E' or 'G', sconda is an estimate of sqrt(||(R**t * R)**(-1)||_1). It is computed using ?pocon. It holds n**(-1/4) * sconda ≤ ||R**(-1)||_2 ≤ n**(1/4) * sconda, where R is the triangular factor from the QRF of A. However, if R is truncated and the numerical rank is determined to be strictly smaller than n, sconda is returned as -1, indicating that the smallest singular values might be lost.

If full SVD is needed, the following two condition numbers are useful for the analysis of the algorithm. They are provied for a user who is familiar with the details of the method.

work(4) = an estimate of the scaled condition number of the triangular factor in the first QR factorization.

work(5) = an estimate of the scaled condition number of the triangular factor in the second QR factorization.

The following two parameters are computed if jobt = 'T'. They are provided for a user who is familiar with the details of the method.

work(6) = the entropy of A**t*A :: this is the Shannon entropy of diag(A**t*A) / Trace(A**t*A) taken as point in the probability simplex.

work(7) = the entropy of A*A**t.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

ENUM_SVDJH_A

An enumeration that specifies the level of accuracy of the SVD computation.

ID	Description
SVDJHA_C	Computation as with 'C' with an additional estimate of the condition number. It provides a realistic error bound.
SVDJHA_E	Computation as with SVDJHA_C' with an additional estimate of the condition number. It provides a realistic error bound.

© 2000–2025, MetaQuotes Ltd.

ID	Description
SVDJHA_F	Higher accuracy than the SVDJHA_C option.
SVDJHA_G	Computation as with SVDJHA_F with an additional estimate of the condition number.
SVDJHA_A	Small singular values are the noise and the matrix is treated as numerically rank deficient.
SVDJHA_R	Similar as in SVDJHA_A, but more accuracy.

ENUM_SVDJH_U

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJHU_U	N columns of U are returned in the array U.
SVDJHU_F	Full set of M left singular vectors is returned in the array U.
SVDJHU_N	U is not computed.

ENUM_SVDJH_V

An enumeration defining how right singular vectors should be computed.

ID	Description
SVDJHV_V	N columns of V are returned in the array V
SVDJHV_J	N columns of V are returned in the array V, but they are computed as the product of Jacobi rotations
SVDJHV_N	V is not computed

ENUM_SVDJH_R

An enumeration that defines the range of values to be computed.

ID	Description
SVDJHR_N	Do not kill small columns of c*A.
SVDJHR_R	RESTRICTED range for sigma(c*A). This option is recommended.

ENUM_SVDJH_T

An enumeration that specifies whether a matrix should be transposed if it is square.

© 2000–2025, MetaQuotes Ltd.

ID	Description
SVDJHT_T	Transpose if entropy test indicates possibly faster convergence of Jacobi process.
SVDJHT_N	Do not use transposition. Do not speculate.

ENUM_SVDJH_P

An enumeration that specifies the possibility of structured perturbations to remove denormalized values.

ID	Description
SVDJHP_P	Introduce perturbation.
SVDJHP_N	Do not perturb.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionJacobiLow

Singular Value Decomposition, Jacobi low level algorithm (LAPACK function GESVJ). The method computes small singular values and their singular vectors with much greater accuracy than other SVD routines in certain cases.

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vector&        S,               // vector of computed singular values
   matrix&        U,               // matrix of computed left vectors U
   matrix&        V,               // matrix of computed left vectors V
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vectorf&       S,               // vector of computed singular values
   matrixf&       U,               // U matrix of computed left vectors
   matrixf&       V,               // V matrix of computed left vectors
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U   jobu,            // how to compute left vectors
   ENUM_SVDJH_V   jobv,            // how to compute right vectors
   double         ctol,            // threshold for convergence if jobu='C'
   ulong          mv,              // number of first rows of matrix V if jobv='A'
   vector&        S,               // vector of computed singular values
   matrixc&       U,               // U matrix of computed left vectors
   matrixc&       V,               // V matrix of computed left vectors
   vector&        work_results     // additional computation results
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionJacobiLow(
   ENUM_SVDJH_U    jobu,                  // how to compute left vectors
   ENUM_SVDJH_V    jobv,                  // how to compute right vectors
```

© 2000–2025, MetaQuotes Ltd.

```
   double          ctol,                  // threshold for convergence if jobu='C'
   ulong           mv,                    // number of first rows of matrix V if jobv
   vectorf&        singular_values,       // vector of computed singular values S
   matrixcf&       u,                     // matrix of computed left vectors U
   matrixcf&       v,                     // matrix of computed left vectors V
   vectorf&        work_results           // additional computation results
   );
```

Parameters

jobu

[in] ENUM_SVDJL_U enumeration value that determines how the left singular vectors should be computed.

jobv

[in] ENUM_SVDJL_V enumeration value defining how the right singular vectors should be computed.

ctol

[in] Convergence threshold if jobu=SVDJLU_C. For other values of 'jobu' the parameter is ignored.

mv

[in] Number of rows of matrix V to be computed if jobv=SVDJLV_A. For other values of 'jobv' the parameter is ignored.

S

[out] Vector of singular values.

Depending on the value scale = work(1), where scale is the scaling factor:

if scale = 1, S(1:n) contains the computed singular values of a. During the computation, sva contains the Euclidean column norms of the iterated matrices in the array a.

if scale ≠ 1, the singular values of a are scale*S(1:n), and this factored representation is due to the fact that some of the singular values of a might underflow or overflow.

U

[out] Matrix of left singular vectors.

V

[out] Matrix of right singular vectors (non-transposed).

work_results

[out] Vector consisting of 7 statistics obtained as a result of the computation.

work(1) = scale is the scaling factor such that scale*S(1:n) are the computed singular values of A. See the description of S).

work(2) is the number of the computed nonzero singular value.

work(3) is the number of the computed singular values that are larger than the underflow threshold.

work(4) is the number of sweeps of Jacobi rotations needed for numerical convergence.

© 2000–2025, MetaQuotes Ltd.

work(5) = max_{i.NE.j} |COS(A(:,i),A(:,j))| in the last sweep. This is useful information in cases when ?gesvj did not converge, as it can be used to estimate whether the output is still useful and for post festum analysis.

work(6) is the largest absolute value over all sines of the Jacobi rotation angles in the last sweep. It can be useful in a post festum analysis.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The number of matrix rows must not be less than the number of columns.

ENUM_SVDJL_U

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJLU_U	The left singular vectors corresponding to the nonzero singular values are computed and returned in the leading columns of A
SVDJLU_C	Analogous to SVDJLU_U, except that user can control the level of numerical orthogonality of the computed left singular vectors.
SVDJLU_N	The matrix U is not computed.

ENUM_SVDJL_V

An enumeration defining how right singular vectors should be computed.

ID	Description
SVDJLV_V	The matrix V is computed
SVDJLV_A	The Jacobi rotations are applied to the MV-by-N matrix V.
SVDJLV_N	The matrix V is not computed.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagDC

Singular Value Decomposition, divide-and-conquer algorithm for bidiagonal matrices (LAPACK function BDSDC).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagDC(
   ENUM_SVDBIDIAG_Z     jobz,        // how to compute left vectors
   ENUM_BLAS_RANGE      range,       // subset of computed singular values
   double               lower,       // lower limit of the subset
   double               upper,       // upper limit of the subset
   vector&              S,           // vector of computed singular values
   matrix&              U,           // U matrix of computed left vectors
   matrix&              VT           // VT transposed matrix of right vectors
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagDC(
   ENUM_SVDBIDIAG_Z     jobz,        // how to compute left vectors
   ENUM_BLAS_RANGE      range,       // subset of computed singular values
   double               lower,       // lower limit of the subset
   double               upper,       // upper limit of the subset
   vectorf&             S,           // vector of computed singular values
   matrixf&             U,           // U matrix of computed left vectors
   matrixf&             VT           // VT transposed matrix of right vectors
   );
```

Parameters

jobz

[in] ENUM_SVDBIDIAG_Z enumeration value that determines how the left singular vectors should be computed.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

© 2000–2025, MetaQuotes Ltd.

V

[out] Transposed matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the jobz and range parameters.

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

A bidiagonal matrix is a square matrix with non-zero main diagonal and one of the sub-diagonals.

Upper bidiagonal matrix

```
[[x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x],
 [0, 0, 0, 0, x]]
```

Lower bidiagonal matrix

```
[[x, 0, 0, 0, 0],
 [x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x]]
```

ENUM_SVDBIDIAG_Z

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJOBZ_V	Compute singular values and singular vectors.
SVDJOBZ_N	Compute singular values only.

ENUM_BLAS_RANGE

© 2000–2025, MetaQuotes Ltd.

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagBisect

Singular Value Decomposition, bisection algorithm for bidiagonal matrices (LAPACK function BDSVDX).

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagBisect(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vector&             S,          // vector of computed singular values
   matrix&             U,          // U matrix of computed left vectors
   matrix&             VT          // VT transposed matrix of right vectors
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagBisect(
   ENUM_SVDBIDIAG_Z    jobz,       // how to compute left vectors
   ENUM_BLAS_RANGE     range,      // subset of computed singular values
   double              lower,      // lower limit of the subset
   double              upper,      // upper limit of the subset
   vectorf&            S,          // vector of computed singular values
   matrixf&            U,          // U matrix of computed left vectors
   matrixf&            VT          // VT transposed matrix of right vectors
   );
```

Parameters

jobz

[in] ENUM_SVDBIDIAG_Z enumeration value that determines how the left singular vectors should be computed.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable singular values and vectors.

lower

[in] The lower limit of singular values subset; specified depending on the value of the range parameter.

upper

[in] The upper limit of singular values subset; specified depending on the value of the range parameter.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

© 2000–2025, MetaQuotes Ltd.

V

[out] Transposed matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

When BLASRANGE_A is set, all singular values are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those singular values (and their vectors) that fall within the range of real values specified by the 'lower' and 'upper' parameters are computed.

With the BLASRANGE_I value, only those singular values (and their vectors) that fall within the range of integer indices specified by the 'lower' and 'upper' parameters are computed. For example, with lower=0 and upper=2, only the first three singular values are computed.

A bidiagonal matrix is a square matrix with non-zero main diagonal and one of the sub-diagonals.

Upper bidiagonal matrix

```
[[x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x],
 [0, 0, 0, 0, x]]
```

Lower bidiagonal matrix

```
[[x, 0, 0, 0, 0],
 [x, x, 0, 0, 0],
 [0, x, x, 0, 0],
 [0, 0, x, x, 0],
 [0, 0, 0, x, x]]
```

ENUM_SVDBIDIAG_Z

An enumeration defining how left singular vectors should be computed.

ID	Description
SVDJOBZ_V	Compute singular values and singular vectors.
SVDJOBZ_N	Compute singular values only.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

See also

SingularValueDecompositionDC, SingularValueDecompositionQR

© 2000–2025, MetaQuotes Ltd.

SingularValueDecompositionBidiagQR

Computes the singular value decomposition of a general matrix that has been reduced to bidiagonal form by the method ReduceToBidiagonal. LAPACK function BDSQR.

Computing for type matrix<double>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // orthogonal matrix Q
   matrix&          PT,         // transposed matrix P
   vector&          S,          // vector of computed singular values
   matrix&          U,          // matrix U of computed left vectors
   matrix&          VT          // transposed matrix V of right vectors
   );
```

Computing for type matrix<float>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // orthogonal matrix Q
   matrix&          PT,         // transposed matrix P
   vectorf&         S,          // vector of computed singular values
   matrixf&         U,          // matrix U of computed left vectors
   matrixf&         VT          // transposed matrix V of right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrix&          Q,          // orthogonal matrix Q
   matrix&          PH,         // Hermitian conjugated matrix P
   vector           S,          // vector of computed singular values
   matrixc          U,          // matrix U of computed left vectors
   matrixc          VH          // Hermitian conjugated matrix V of right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SingularValueDecompositionBidiagQR(
   matrixf&         Q,          // orthogonal matrix Q
   matrixf&         PH,         // Hermitian conjugated matrix P
   vectorf&         S,          // vector of computed singular values
   matrixcf&        U,          // matrix U of computed left vectors
   matrixcf&        VH          // Hermitian conjugated matrix V of right vectors
   );
```

Parameters

Q

[in] Orthogonal matrix Q produced by method ReflectBidiagonalToQP. If matrix Q has zero size, then left vectors U are not calculated.

PT

© 2000–2025, MetaQuotes Ltd.

[in] Transposed (or hermitian conjugated) matrix P produced by method ReflectBidiagonalToQP. If matrix PT has zero size, then right vectors VT are not calculated.

S

[out] Vector of singular values.

U

[out] Matrix of left singular vectors.

VT

[out] Matrix of right singular vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

BDSQR computes the singular values and, optionally, the right and/or left singular vectors from the singular value decomposition (SVD) of a N-by-N (upper or lower) bidiagonal matrix B using the implicit zero-shift QR algorithm. The SVD of B has the form

B = Q * S * P**T

where S is the diagonal matrix of singular values, Q is an orthogonal matrix of left singular vectors, and P is an orthogonal matrix of right singular vectors. If left singular vectors are requested, this subroutine actually returns U*Q instead of Q, and, if right singular vectors are requested, this subroutine returns P**T*VT instead of P**T, for given real input matrices U and VT. When U and VT are the orthogonal matrices that reduce a general matrix A to bidiagonal form: A = U*B*VT, as computed by GEBRD, then

A = (U*Q) * S * (P**T*VT)

is the SVD of A.

© 2000–2025, MetaQuotes Ltd.

Eigen Values

The section features functions for computing eigenvalues and eigenvectors. It describes methods for solving standard linear algebra problems using the LAPACK library algorithms. These functions are efficient for matrix analysis, diagonalization, system stabilization, and other tasks.

- **EigenSolver**: The function is designed to compute the eigenvalues and eigenvectors of an arbitrary square matrix using the classical algorithm represented by the GEEV LAPACK function. This method is applied to a wide range of matrices, allowing the decomposition of matrices into their eigenvalues and eigenvectors.
- **EigenSymmetricDC**: The function for computing eigenvalues and eigenvectors of symmetric or Hermitian matrices using the divide-and-conquer algorithm. The LAPACK functions SYEVD and HEEVD enable the efficient handling of symmetric or Hermitian matrices, providing faster and more accurate processing of such matrices.

Function	Action
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (LAPACK function GEEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (LAPACK function GEEVX).
EigenSolverSchur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (LAPACK function GEES). See also Schur decomposition.
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (LAPACK function GGEV).
EigenSolver2X	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (LAPACK function GGEVX). Both matrices must be the same size.
EigenSolver2Schur	Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (LAPACK function GGES).
EigenSolver2Blocked	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (LAPACK function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.

© 2000–2025, MetaQuotes Ltd.

Function	Action
EigenSolver2SchurBlocked	Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (LAPACK function GGES3).
EigenHessenbergSchurQ	Computes the eigenvalues of a Hessenberg matrix H and the matrices T and Z from the Schur decomposition H = Z T Z**T, where T is an upper quasi-triangular matrix (the Schur form), and Z is the orthogonal matrix of Schur vectors. LAPACK function HSEQR. See also Schur decomposition.
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (LAPACK functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the QR algorithm (LAPACK functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (LAPACK functions SYEVX, HEEVX).
EigenSymmetricDC2s	Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal. If eigenvectors are desired, it uses a divide and conquer algorithm (LAPACK functions SYEVD_2STAGE, HEEVD_2STAGE).
EigenSymmetricQR2s	Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal (LAPACK functions SYEV_2STAGE, HEEV_2STAGE).
EigenSymmetricRobust2s	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal then using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK functions SYEVR_2STAGE, HEEVR_2STAGE).

© 2000-2025, MetaQuotes Ltd.

Function	Action
EigenSymmetricBisect2s	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal then using the bisection algorithm (LAPACK functions SYEVX_2STAGE, HEEVX_2STAGE).
EigenSymmetric2DC	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x. LAPACK functions SYGVD, HEGVD.
EigenSymmetric2QR	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x. LAPACK functions SYGV, HEGV.
EigenSymmetric2Bisect	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x. LAPACK functions SYGVX, HEGVX.
EigenTridiagonalDC	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (LAPACK function STEVD).
EigenTridiagonalQR	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (LAPACK function STEV).
EigenTridiagonalRobust	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK function STEVR).
EigenTridiagonalBisect	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the bisection algorithm (LAPACK function STEVX).
EigenTridiagonalQL	Compute all eigenvalues of a symmetric tridiagonal matrix using the Pal-Walker-Kahan variant of the QL or QR algorithm (LAPACK function STERF).
EigenTridiagonalDCQ	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (LAPACK function STEDC).
EigenTridiagonalQRQ	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (LAPACK function STEQR).

© 2000–2025, MetaQuotes Ltd.

Function	Action
EigenTridiagonalPosDefQ	Compute eigenvalues and eigenvectors of a symmetric positive definite (положительно определённая) tridiagonal matrix using the QR algorithm (LAPACK function PTEQR).

© 2000–2025, MetaQuotes Ltd.

General Matrices

Functions for calculating eigenvalues and eigenvectors of a square matrix using classical algorithms. It provides various methods for working with both real and complex matrices, allowing you to solve linear algebra problems with a choice of methods for calculating eigenvectors.

Function	Action
EigenSolver	Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (LAPACK function GEEV).
EigenSolverX	Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (LAPACK function GEEVX).
EigenSolverSchur	Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (LAPACK function GEES). See also Schur decomposition.
EigenSolver2	Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (LAPACK function GGEV).
EigenSolver2X	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (LAPACK function GGEVX). Both matrices must be the same size.
EigenSolver2Schur	Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (LAPACK function GGES).
EigenSolver2Blocked	Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (LAPACK function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.
EigenSolver2SchurBlocked	Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (LAPACK function GGES3).
EigenHessenbergSchurQ	Computes the eigenvalues of a Hessenberg matrix H and the matrices T and Z from the Schur decomposition H = Z T Z**T, where T is an upper quasi-triangular matrix (the Schur form), and Z is

© 2000–2025, MetaQuotes Ltd.

Function	Action
	the orthogonal matrix of Schur vectors. LAPACK function HSEQR. See also Schur decomposition.
EigenVectorsTriangularZ	Computes eigenvectors of an real upper quasi-triangular or complex upper triangular matrix computed by EigenHessenbergSchurQ or EigenSolverSchur. A = Q * T * Q**H, where T is an upper quasi-triangular matrix (the Schur form), and Q is the orthogonal matrix of Schur vectors. LAPACK function TREVC.
EigenVectorsTriangularZBlocked	Computes eigenvectors of an real upper quasi-triangular or complex upper triangular matrix computed by EigenHessenbergSchurQ or EigenSolverSchur. A = Q * T * Q**H, where T is an upper quasi-triangular matrix (the Schur form), and Q is the orthogonal matrix of Schur vectors. LAPACK function TREVC3. This is the block version (OpenBLAS level 3) of TREVC. Faster but not so accurate.

© 2000–2025, MetaQuotes Ltd.

EigenSolver

Compute eigenvalues and eigenvectors of a regular square matrix using the classical algorithm (LAPACK function GEEV).

Computing for type matrix<double>

```
bool  matrix::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right eigenvecto
   vectorc&              eigen_values,         // vector of computed eigenvalues
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right eigenvecto
   vectorcf&             eigen_values,         // vector of computed eigenvalues
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right vectors
   vectorc&              eigen_values,         // vector of computed eigenvalues
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver(
   ENUM_EIG_VECTORS      jobv,                 // compute left and right vectors
   vectorcf&             eigen_values,         // vector of computed eigenvalues
   matrixcf&             left_eigenvectors,    // matrix of computed left vectors
   matrixcf&             right_eigenvectors    // matrix of computed right vectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

EV

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

© 2000–2025, MetaQuotes Ltd.

right_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

If EIGVECTORS_N is set, the left and right vectors are not computed. Only eigenvalues are computed.

With EIGVECTORS_L, only left eigenvectors are computed, right eigenvectors are not computed.

When EIGVECTORS_R is set, only the right eigenvectors are computed, the left vectors are not computed.

With EIGVECTORS_LR, the left and right eigenvectors are computed, Eigenvalues are always computed.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

ENUM_EIG_VECTORS

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are calculated, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

EigenSolverX

Compute eigenvalues and eigenvectors of a regular square matrix in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (LAPACK function GEEVX).

Computing for type matrix<double>

```
bool  matrix::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorc&              eigen_values,         // vector of computed eigenvalues
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   matrix&               schur_matrix,         // balanced matrix in Schur form
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vector&               scale,                // details of permutations and scali
   double&               ab_norm,              // 1-norm of balanced matrix
   vector&               rconde,               // vector of reciprocal condition nu
   vector&               rcondv                // vector of reciprocal condition nu
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorcf&             eigen_values,         // vector of computed eigenvalues
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   matrixf&              schur_matrix,         // balanced matrix in Schur form
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vectorf&              scale,                // details of permutations and scali
   float&                ab_norm,              // 1-norm of balanced matrix
   vectorf&              rconde,               // vector of reciprocal condition nu
   vectorf&              rcondv                // vector of reciprocal condition nu
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolverX(
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorc&              eigen_values,         // vector of computed eigenvalues
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&              left_eigenvectors,       // matrix of computed left vectors
   matrixc&              right_eigenvectors       // matrix of computed right vectors
   matrixc&              schur_matrix,            // balanced matrix in Schur form
   long&                 ilo,                     // subscript of balanced matrix
   long&                 ihi,                     // superscript of balanced matrix
   vector&               scale,                   // details of permutations and scali
   double&               ab_norm,                 // 1-norm of balanced matrix
   vector&               rconde,                  // vector of reciprocal condition nu
   vector&               rcondv                   // vector of reciprocal condition nu
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolverX(
   ENUM_EIG_BALANCE      balance,                 // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                    // determines computation of right a
   ENUM_EIG_SENSE        sense,                   // determines computation of recipro
   vectorcf&             eigen_values,            // vector of computed eigenvalues
   matrixcf&             left_eigenvectors,       // matrix of computed left vectors
   matrixcf&             right_eigenvectors       // matrix of computed right vectors
   matrixcf&             schur_matrix,            // balanced matrix in Schur form
   long&                 ilo,                     // subscript of balanced matrix
   long&                 ihi,                     // superscript of balanced matrix
   vectorf&              scale,                   // details of permutations and scali
   float&                ab_norm,                 // 1-norm of balanced matrix
   vectorf&              rconde,                  // vector of reciprocal condition nu
   vectorf&              rcondv                   // vector of reciprocal condition nu
   );
```

Parameters

balance

[in] Value from the ENUM_EIG_BALANCE enumeration which determines the need and method for balancing the input matrix; it is used to improve the conditioning of the eigenvalues and eigenvectors.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

sense

[in] Value from the ENUM_EIG_SENSE enumeration determining the need to compute reciprocal condition numbers.

eigen_values

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

© 2000-2025, MetaQuotes Ltd.

[out] Matrix of right eigenvectors.

schur_matrix

[out] Balanced matrix in Schur form; the matrix is not filled if neither left nor right eigenvectors are computed.

ilo

[out] Subscript of the balanced matrix; the matrix is not filled if no balancing is applied.

ihi

[out] Superscript of the balanced matrix; the matrix is not filled if no balancing is applied.

scale

[out] Vector of details of permutations and scaling when balancing the input matrix.

Details of the permutations and scaling factors applied when balancing A.

If P(j) is the index of the row and column interchanged with row and column j, and D(j) is the scaling factor applied to row and column j, then

scale(j) = P(j), for j = 1,...,ilo-1

= D(j), for j = ilo,...,ihi

= P(j) for j = ihi+1,..., n.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

ab_norm

[out] 1-norm of the balanced matrix (the maximum of the sum of absolute values of elements in any of the matrix columns).

rconde

[out] Vector of reciprocal condition numbers for each eigenvalue; it is computed if the 'sense' parameter is set to 'E' or 'B'.

rcondv

[out] Vector of reciprocal condition numbers for each eigenvector; it is computed if the 'sense' parameter is set to 'V' or 'B.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Input matrix balancing depends on the value of the '*balance*' parameter.

ENUM_EIG_BALANCE

An enumeration defining the need to compute eigenvectors.

© 2000–2025, MetaQuotes Ltd.

ID	Description
EIGBALANCE_N	Do not diagonally scale or permute
EIGBALANCE_P	Perform permutations to make the matrix more nearly upper triangular. Do not diagonally scale
EIGBALANCE_S	Diagonally scale the matrix. Do not permute
EIGBALANCE_B	Both diagonally scale and permute

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

ENUM_EIG_SENSE

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGSENSE_N	None of reciprocal condition numbers are computed
EIGSENSE_E	Computed for eigenvalues only
EIGSENSE_V	Computed for right eigenvectors only
EIGSENSE_B	Computed for eigenvalues and right eigenvectors

© 2000–2025, MetaQuotes Ltd.

EigenSolverSchur

Compute eigenvalues, upper triangular matrix in Schur form, and matrix of Schur vectors (LAPACK function GEES). See also Schur decomposition.

Computing for type matrix<double>

```
bool  matrix::EigenSolverSchur(
   ENUM_EIG_SCHUR        jobvs,                  // method to compute Schur vectors
   vectorc&              eigen_values,           // vector of computed eigenvalues
   matrix&               schur_matrix,           // matrix in Schur form
   matrix&               schur_vectors           // matrix of Schur vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolverSchur(
   ENUM_EIG_SCHUR        jobvs,                  // method to compute Schur vectors
   vectorcf&             eigen_values,           // vector of computed eigenvalues
   matrixf&              schur_matrix,           // matrix in Schur form
   matrixf&              schur_vectors           // matrix of Schur vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolverSchur(
   ENUM_EIG_SCHUR        jobvs,                  // method to compute Schur vectors
   vectorc&              eigen_values,           // vector of computed eigenvalues
   matrixc&              schur_matrix,           // matrix in Schur form
   matrixc&              schur_vectors           // matrix of Schur vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolverSchur(
   ENUM_EIG_SCHUR        jobvs,                  // method to compute Schur vectors
   vectorcf&             eigen_values,           // vector of computed eigenvalues
   matrixcf&             schur_matrix,           // matrix in Schur form
   matrixcf&             schur_vectors           // matrix of Schur vectors
   );
```

Parameters

jobvs

[in] Value from the ENUM_EIG_SCHUR enumeration, which defines the method for computing Schur vectors.

eigen_values

[out] Vector of eigenvalues.

schur_matrix

[out] Upper triangular Schur matrix (Schur form for the input matrix).

© 2000–2025, MetaQuotes Ltd.

schur_vectors

[out] Matrix of Schur vectors; it is not computed if the *jobvs* parameter is set to N.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the *jobvs* parameter values.

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

ENUM_EIG_SCHUR

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGSCHUR_N	Schur vectors are not computed
EIGSCHUR_V	Schur vectors are computed

© 2000–2025, MetaQuotes Ltd.

EigenSolver2

Compute generalized eigenvalues and eigenvectors for a pair of ordinary square matrices (LAPACK function GGEV). Both matrices must be the same size.

Computing for type matrix<double>

```
bool  matrix::EigenSolver2(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorf&              beta,                 // vector of eigenvalue divisors
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver2(
   matrixc&              B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorc&              alpha,                // vector of computed eigenvalues
   vectorc&              beta,                 // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver2(
   matrixcf&             B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorcf&             beta,                 // vector of eigenvalue divisors
   matrixcf&             left_eigenvectors,    // matrix of computed left vectors
   matrixcf&             right_eigenvectors    // matrix of computed right vectors
   );
```

Parameters

© 2000–2025, MetaQuotes Ltd.

B

[in] The second matrix in the pair.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigen value divisors.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

A generalized eigenvalue for a pair of matrices (A,B) is a scalar lambda or a ratio alpha/beta = lambda, such that A - lambda*B is singular. It is usually represented as the pair (alpha,beta), as there is a reasonable interpretation for beta=0, and even for both being zero.

The right eigenvector v(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies

A * v(j) = lambda(j) * B * v(j).

The left eigenvector u(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies

u(j)**H * A = lambda(j) * u(j)**H * B .

© 2000–2025, MetaQuotes Ltd.

where u(j)**H is the conjugate-transpose of u(j).

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

© 2000–2025, MetaQuotes Ltd.

EigenSolver2X

Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices in Expert mode, i.e. with the ability to influence the computation algorithm and the ability to obtain accompanying computation data (LAPACK function GGEVX). Both matrices must be the same size.

Optionally, it also computes a balancing transformation to improve the conditioning of the eigenvalues and eigenvectors (ILO, IHI, LSCALE, RSCALE, ABNRM, and BBNRM), reciprocal condition numbers for the eigenvalues (RCONDE), and reciprocal condition numbers for the right eigenvectors (RCONDV).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2X(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   matrix&               schur_matrix1,        // the first part of the real Schur
   matrix&               schur_matrix2,        // the second part of the real Schu
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vector&               lscale,               // details of the permutations and s
   vector&               rscale,               // details of the permutations and s
   double&               ab_norm,              // one-norm of balanced input matri
   double&               bb_norm,              // one-norm of balanced second matr
   vector&               rconde,               // vector of reciprocal condition nu
   vector&               rcondv                // vector of reciprocal condition nu
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2X(
   matrixf&              B,                    // second matrix in the pair
   ENUM_EIG_BALANCE      balance,              // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                 // determines computation of right a
   ENUM_EIG_SENSE        sense,                // determines computation of recipro
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorf&              beta,                 // vector of eigenvalue divisors
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   matrixf&              schur_matrix1,        // the first part of the real Schur
   matrixf&              schur_matrix2,        // the second part of the real Schu
   long&                 ilo,                  // subscript of balanced matrix
   long&                 ihi,                  // superscript of balanced matrix
   vectorf&              lscale,               // details of the permutations and s
```

© 2000-2025, MetaQuotes Ltd.

```
   vectorf&              rscale,                 // details of the permutations and s
   float&                ab_norm,                // one-norm of balanced input matrix
   float&                bb_norm,                // one-norm of balanced second matri
   vectorf&              rconde,                 // vector of reciprocal condition nu
   vectorf&              rcondv                  // vector of reciprocal condition nu
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSolver2X(
   matrixc&              B,                      // second matrix in the pair
   ENUM_EIG_BALANCE      balance,                // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                   // determines computation of right a
   ENUM_EIG_SENSE        sense,                  // determines computation of recipro
   vectorc&              alpha,                  // vector of computed eigenvalues
   vectorc&              beta,                   // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,      // matrix of computed left vectors
   matrixc&              right_eigenvectors      // matrix of computed right vectors
   matrixc&              schur_matrix1,          // the first part of the real Schur
   matrixc&              schur_matrix2,          // the second part of the real Schur
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vector&               lscale,                 // details of the permutations and s
   vector&               rscale,                 // details of the permutations and s
   double&               ab_norm,                // one-norm of balanced input matrix
   double&               bb_norm,                // one-norm of balanced second matri
   vector&               rconde,                 // vector of reciprocal condition nu
   vector&               rcondv                  // vector of reciprocal condition nu
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver2X(
   matrixcf&             B,                      // second matrix in the pair
   ENUM_EIG_BALANCE      balance,                // input matrix balancing method
   ENUM_EIG_VECTORS      jobv,                   // determines computation of right a
   ENUM_EIG_SENSE        sense,                  // determines computation of recipro
   vectorcf&             alpha,                  // vector of computed eigenvalues
   vectorf&              beta,                   // vector of eigenvalue divisors
   matrixcf&             left_eigenvectors,      // matrix of computed left vectors
   matrixcf&             right_eigenvectors      // matrix of computed right vectors
   matrixcf&             schur_matrix1,          // the first part of the real Schur
   matrixcf&             schur_matrix2,          // the second part of the real Schur
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vectorf&              lscale,                 // details of the permutations and s
   vectorf&              rscale,                 // details of the permutations and s
   float&                ab_norm,                // one-norm of balanced input matrix
```

© 2000-2025, MetaQuotes Ltd.

```
   float&               bb_norm,              // one-norm of balanced second matri
   vectorf&             rconde,               // vector of reciprocal condition nu
   vectorf&             rcondv                // vector of reciprocal condition nu
   );
```

Parameters

B

[in] The second matrix in the pair.

balance

[in] Value from the ENUM_EIG_BALANCE enumeration which determines the need and method for balancing the input matrix; it is used to improve the conditioning of the eigenvalues and eigenvectors.

jobv

[in] Value from the ENUM_EIG_VECTORS enumeration which determines the method for computing left and right eigenvectors.

sense

[in] Value from the ENUM_EIG_SENSE enumeration which determines the need to compute reciprocal condition numbers.

eigen_values

[out] Vector of eigenvalues.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

schur_matrix1, schur_matrix2

[out] 2 parts of balanced matrix in Schur form; the matrix is not filled if neither left nor right eigenvectors are computed.

ilo

[out] Subscript of the balanced matrix; the matrix is not filled if no balancing is applied.

ihi

[out] Superscript of the balanced matrix; the matrix is not filled if no balancing is applied.

lscale

[out] Vector contains details of the permutations and scaling factors applied to the left side of A and B.

If PL(j) is the index of the row interchanged with row j, and DL(j) is the scaling factor applied to row j, then

lscale(j) = PL(j), for j = 1,..., ilo-1

= DL(j), for j = ilo,...,ihi

= PL(j) for j = ihi+1,..., n.

© 2000–2025, MetaQuotes Ltd.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

rscale

[out] Vector contains details of the permutations and scaling factors applied to the right side of A and B.

If PR(j) is the index of the column interchanged with column j, and DR(j) is the scaling factor applied to column j, then

rscale(j) = PR(j), for j = 1,..., ilo-1

= DR(j), for j = ilo,...,ihi

= PR(j) for j = ihi+1,..., n.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

ab_norm

[out] One-norm of the balanced input matrix (the maximum of the sum of absolute values of elements in any of the matrix columns).

bb_norm

[out] One-norm of balanced second matrix B.

rconde

[out] Vector of reciprocal condition numbers for each eigenvalue; it is computed if the 'sense' parameter is set to 'E' or 'B'.

rcondv

[out] Vector of reciprocal condition numbers for each eigenvector; it is computed if the 'sense' parameter is set to 'V' or 'B'.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Input matrices balancing depends on the value of the *balance* parameter.

ENUM_EIG_BALANCE

An enumeration that specifies whether the matrices should be balanced.

ID	Description
EIGBALANCE_N	Do not diagonally scale or permute.
EIGBALANCE_P	Perform permutations to make the matrix more nearly upper triangular. Do not diagonally scale.
EIGBALANCE_S	Diagonally scale the matrix. Do not permute.
EIGBALANCE_B	Both diagonally scale and permute.

© 2000–2025, MetaQuotes Ltd.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

ENUM_EIG_SENSE

An enumeration determining the need to compute reciprocal condition numbers.

ID	Description
EIGSENSE_N	None of reciprocal condition numbers are computed.
EIGSENSE_E	Computed for eigenvalues only.
EIGSENSE_V	Computed for right eigenvectors only.
EIGSENSE_B	Computed for eigenvalues and right eigenvectors.

© 2000–2025, MetaQuotes Ltd.

EigenSolver2Schur

Compute a pair of ordinary square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (LAPACK function GGES).

Computes the generalized eigenvalues, the generalized real/complex Schur form (S,T), optionally, the left and/or right matrices of Schur vectors (vsl and vsr) for a pair of n-by-n real/complex nonsymmetric matrices (A,B). This gives the generalized Schur factorization:

(A,B) = (vsl*S *vsrH, vsl*T*vsrH)

Optionally, it also orders the eigenvalues so that a selected cluster of eigenvalues appears in the leading diagonal blocks of the upper quasi-triangular matrix S and the upper triangular matrix T. The leading columns of vsl and vsr then form an orthonormal/unitary basis for the corresponding left and right eigenspaces (deflating subspaces).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2Schur(
   matrix&                 B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vector&                 alpha,     // vector of computed eigenvalues
   vector&                 beta,      // vector of eigenvalue divisors
   matrix&                 schur_s,   // matrix S in Schur form
   matrix&                 schur_t,   // matrix T in Schur form
   matrix&                 vsl,       // matrix of left Schur vectors vsl
   matrix&                 vsr        // matrix of right Schur vectors vsr
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2Schur(
   matrixf&                B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vectorcf&               alpha,     // vector of computed eigenvalues
   vectorf&                beta,      // vector of eigenvalue divisors
   matrixf&                schur_s,   // matrix S in Schur form
   matrixf&                schur_t,   // matrix T in Schur form
   matrixf&                vsl,       // matrix of left Schur vectors vsl
   matrixf&                vsr        // matrix of right Schur vectors vsr
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2Schur(
   matrixc&                B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vectorc&                alpha,     // vector of computed eigenvalues
   vectorc&                beta,      // vector of eigenvalue divisors
   matrixc&                schur_s,   // matrix S in Schur form
   matrixc&                schur_t,   // matrix T in Schur form
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixc&                vsl,       // matrix of left Schur vectors vsl
   matrixc&                vsr        // matrix of right Schur vectors vsr
   );
```

Вычисления для типа matrix<complexf>

```
bool  matrixcf::EigenSolver2Schur(
   matrixcf&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,     // method to compute left and right vectors
   vectorcf&               alpha,     // vector of computed eigenvalues
   vectorcf&               beta,      // vector of eigenvalue divisors
   matrixcf&               schur_s,   // matrix S in Schur form
   matrixcf&               schur_t,   // matrix T in Schur form
   matrixcf&               vsl,       // matrix of left Schur vectors vsl
   matrixcf&               vsr        // matrix of right Schur vectors vsr
   );
```

Parameters

B

[out] The second matrix in the pair.

jobvs

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

schur_s

[out] Matrix S, block upper triangular Schur matrix (Schur form for the input matrix).

schur_t

[out] Matrix T, block upper triangular Schur matrix (Schur form for the second matrix in the pair).

vsl

[out] Matrix of left Schur vectors.

vsr

[out] Matrix of right Schur vectors.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the *jobvs* parameter values.

© 2000–2025, MetaQuotes Ltd.

The second matrix in the pair must be the same size as the first (input) one.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

EigenSolverSchur

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

EigenSolver2Blocked

Compute generalized eigenvalues and eigenvectors for a pair of regular square matrices using a block algorithm (LAPACK function GGEV3). Both matrices must be the same size. The method parameters are exactly the same as EigenSolver2.

Computing for type matrix<double>

```
bool  matrix::EigenSolver2Blocked(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorc&              alpha,                // vector of computed eigenvalues
   vector&               beta,                 // vector of eigenvalue divisors
   matrix&               left_eigenvectors,    // matrix of computed left vectors
   matrix&               right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSolver2Blocked(
   matrix&               B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorf&              beta,                 // vector of eigenvalue divisors
   matrixf&              left_eigenvectors,    // matrix of computed left vectors
   matrixf&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2Blocked(
   matrixc&              B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorc&              alpha,                // vector of computed eigenvalues
   vectorc&              beta,                 // vector of eigenvalue divisors
   matrixc&              left_eigenvectors,    // matrix of computed left vectors
   matrixc&              right_eigenvectors    // matrix of computed right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSolver2(
   matrixcf&             B,                    // second matrix in the pair
   ENUM_EIG_VECTORS      jobv,                 // method to compute right and left
   vectorcf&             alpha,                // vector of computed eigenvalues
   vectorcf&             beta,                 // vector of eigenvalue divisors
   matrixcf&             left_eigenvectors,    // matrix of computed left vectors
   matrixcf&             right_eigenvectors    // matrix of computed right vectors
   );
```

Parameters

© 2000–2025, MetaQuotes Ltd.

B

[out] The second matrix in the pair.

jobv

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

left_eigenvectors

[out] Matrix of left eigenvectors.

righeft_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the value of the jobv parameter.

A generalized eigenvalue for a pair of matrices (A,B) is a scalar lambda or a ratio alpha/beta = lambda, such that A - lambda*B is singular. It is usually represented as the pair (alpha,beta), as there is a reasonable interpretation for beta=0, and even for both being zero.

The right eigenvector v(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies:

A * v(j) = lambda(j) * B * v(j).

The left eigenvector u(j) corresponding to the eigenvalue lambda(j) of (A,B) satisfies:

u(j)**H * A = lambda(j) * u(j)**H * B .

where u(j)**H is the conjugate-transpose of u(j).

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

ENUM_EIG_VECTORS

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.

© 2000–2025, MetaQuotes Ltd.

ID	Description
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

EigenSolver2SchurBlocked

Compute a pair of regular square matrices of generalized eigenvalues, generalized eigenvectors, generalized Schur forms, as well as left and right Schur vectors (LAPACK function GGES3).

Computes the generalized eigenvalues, the generalized real/complex Schur form (S,T), optionally, the left and/or right matrices of Schur vectors (VSL and VSR) for a pair of n-by-n real/complex nonsymmetric matrices (A,B). This gives the generalized Schur factorization:

(A,B) = (vsl*S *vsrH, vsl*T*vsrH)

Optionally, it also orders the eigenvalues so that a selected cluster of eigenvalues appears in the leading diagonal blocks of the upper quasi-triangular matrix S and the upper triangular matrix T. The leading columns of vsl and vsr then form an orthonormal/unitary basis for the corresponding left and right eigenspaces (deflating subspaces).

Computing for type matrix<double>

```
bool  matrix::EigenSolver2SchurBlocked(
   matrix&                B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorc&               alpha,     // vector of computed eigenvalues
   vector&                beta,      // vector of eigenvalue divisors
   matrix&                schur_s,   // matrix S in Schur form
   matrix&                schur_t,   // matrix T in Schur form
   matrix&                vsl,       // matrix of left Schur vectors VSL
   matrix&                vsr        // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<float>

```
bool  matrix::EigenSolver2SchurBlocked(
   matrixf&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorcf&              alpha,     // vector of computed eigenvalues
   vectorf&               beta,      // vector of eigenvalue divisors
   matrixf&               schur_s,   // matrix S in Schur form
   matrixf&               schur_t,   // matrix T in Schur form
   matrixf&               vsl,       // matrix of left Schur vectors VSL
   matrixf&               vsr        // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<complex>

```
bool  matrix::EigenSolver2SchurBlocked(
   matrixc&               B,         // second matrix in the pair
   ENUM_EIG_VECTORS       jobvs,     // method to compute left and right vectors
   vectorc&               alpha,     // vector of computed eigenvalues
   vectorc&               beta,      // vector of eigenvalue divisors
   matrixc&               schur_s,   // matrix S in Schur form
   matrixc&               schur_t,   // matrix T in Schur form
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixc&                vsl,      // matrix of left Schur vectors VSL
   matrixc&                vsr       // matrix of right Schur vectors VSR
   );
```

Computing for type matrix<complexf>

```
bool  matrix::EigenSolver2SchurBlocked(
   matrixcf&               B,        // second matrix in the pair
   ENUM_EIG_VECTORS        jobvs,    // method to compute left and right vectors
   vectorcf&               alpha,    // vector of computed eigenvalues
   vectorcf&               beta,     // vector of eigenvalue divisors
   matrixcf&               schur_s,  // matrix S in Schur form
   matrixcf&               schur_t,  // matrix T in Schur form
   matrixcf&               vsl,      // matrix of left Schur vectors VSL
   matrixcf&               vsr       // matrix of right Schur vectors VSR
   );
```

Parameters

B

[in] The second matrix in the pair.

jobvs

[in] ENUM_EIG_VECTORS enumeration value which determines the method for computing left and right eigenvectors.

alpha

[out] Vector of eigenvalues.

beta

[out] Vector of eigenvalue divisors.

schur_s

[out] Matrix S, block upper triangular Schur matrix (Schur form for the input matrix).

schur_t

[out] Matrix T, block upper triangular Schur matrix (Schur form for the second matrix in the pair).

vsl

[out] Matrix of left Schur vectors VSL.

vsr

[out] Matrix of right Schur vectors VSR.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

Computation depends on the *jobvs* parameter values.

© 2000–2025, MetaQuotes Ltd.

The second matrix in the pair must be the same size as the first (input) one.

Real (non-complex) matrices can have a complex solution. Therefore, the vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

EigenSolverSchur

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVECTORS_N	Only eigenvalues are computed, without vectors.
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

EigenHessenbergSchurQ

Computes the eigenvalues of a Hessenberg matrix H and the matrices T and Z from the Schur decomposition H = Z T Z**T, where T is an upper quasi-triangular matrix (the Schur form), and Z is the orthogonal matrix of Schur vectors. Optionally Z may be postmultiplied into an input orthogonal matrix Q so that this routine can give the Schur factorization of a matrix A which has been reduced to the Hessenberg form H by the orthogonal matrix Q:

A = Q*H*Q**T = (QZ)*T*(QZ)**T.

LAPACK function HSEQR. See also Schur decomposition.

Computing for type matrix<double>

```
bool  matrix::EigenHessenbergSchurQ(
   matrix&                Q,                    // orthogonal matrix Q
   vectorc&               eigen_values,         // vector of computed eigenvalues
   matrix&                schur_t,              // matrix T in Schur form
   matrix&                schur_z               // matrix Z of Schur vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenHessenbergSchurQ(
   matrixf&               Q,                    // orthogonal matrix Q
   vectorcf&              eigen_values,         // vector of computed eigenvalues
   matrixf&               schur_t,              // matrix T in Schur form
   matrixf&               schur_z               // matrix Z of Schur vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenHessenbergSchurQ(
   matrixc&               Q,                    // orthogonal matrix Q
   vectorc&               eigen_values,         // vector of computed eigenvalues
   matrixc&               schur_t,              // matrix T in Schur form
   matrixc&               schur_z               // matrix Z of Schur vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenHessenbergSchurQ(
   matrixcf&              Q,                    // orthogonal matrix Q
   vectorcf&              eigen_values,         // vector of computed eigenvalues
   matrixcf&              schur_t,              // matrix T in Schur form
   matrixcf&              schur_z               // matrix Z of Schur vectors
   );
```

Parameters

Q

© 2000–2025, MetaQuotes Ltd.

[in] Orthogonal matrix Q produced by method ReflectHessenbergToQ. Matrix Q can be of zero size, in this case Hessenberg matrix (not original matrix A) will be decomposed. If matrix Q is used, then calculated the original matrix A reduced to Hessenberg form (see ReduceToHessenberg).

eigen_values

[out] Vector of eigenvalues.

schur_t

[out] Upper triangular Schur matrix (Schur form for the input matrix).

schur_z

[out] Matrix of Schur vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

© 2000–2025, MetaQuotes Ltd.

EigenVectorsTriangularZ

Computes eigenvectors of an real upper quasi-triangular or complex upper triangular matrix computed by EigenHessenbergSchurQ or EigenSolverSchur.

A = Q * T * Q**H, where T is an upper quasi-triangular matrix (the Schur form), and Q is the orthogonal matrix of Schur vectors.

LAPACK function TREVC.

Computing for type matrix<double>

```
bool  matrix::EigenVectorsTriangularZ(
   ENUM_EIG_VECTORS      side,                  // compute left and/or right eigenve
   matrix&               schur_z,               // orthogonal matrix of Schur vecto
   matrix&               left_eigenvectors,     // matrix of computed left vectors
   matrix&               right_eigenvectors     // matrix of computed right vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenVectorsTriangularZ(
   ENUM_EIG_VECTORS      side,                  // compute left and/or right eigenve
   matrixf&              schur_z,               // orthogonal matrix of Schur vecto
   matrixf&              left_eigenvectors,     // matrix of computed left vectors
   matrixf&              right_eigenvectors     // matrix of computed right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenVectorsTriangularZ(
   ENUM_EIG_VECTORS      side,                  // compute left and/or right eigenve
   matrixc&              schur_z,               // orthogonal matrix of Schur vecto
   matrixc&              left_eigenvectors,     // matrix of computed left vectors
   matrixc&              right_eigenvectors     // matrix of computed right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenVectorsTriangularZ(
   ENUM_EIG_VECTORS      side,                  // compute left and/or right eigenve
   matrixcf&             schur_z,               // orthogonal matrix of Schur vecto
   matrixcf&             left_eigenvectors,     // matrix of computed left vectors
   matrixcf&             right_eigenvectors     // matrix of computed right vectors
   );
```

Parameters

side

[in] ENUM_EIG_VECTORS enumeration value which determines what the eigenvectors are computed left, right or both.

shur_z

© 2000–2025, MetaQuotes Ltd.

[in] Orthogonal matrix of Schur vectors computed by EigenHessenbergSchurQ or EigenSolverSchur. Can be of zero size. In this case no backtransformation is produced.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The right eigenvector x and the left eigenvector y of T corresponding to an eigenvalue w are defined by:

T*x = w*x, (y**H)*T = w*(y**H)

where y**H denotes the conjugate transpose of y.

The eigenvalues are not input to this routine, but are read directly from the diagonal blocks of T.

This routine returns the matrices X and/or Y of right and left eigenvectors of T, or the products Q*X and/or Q*Y, where Q is an input matrix. If Q is the orthogonal factor that reduces a matrix A to Schur form T, then Q*X and Q*Y are the matrices of right and left eigenvectors of A.

ENUM_EIG_VECTORS

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVECTORS_N	Cannot be used with this method
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

EigenVectorsTriangularZBlocked

Computes eigenvectors of an real upper quasi-triangular or complex upper triangular matrix computed by EigenHessenbergSchurQ or EigenSolverSchur.

A = Q * T * Q**H, where T is an upper quasi-triangular matrix (the Schur form), and Q is the orthogonal matrix of Schur vectors.

LAPACK function TREVC3. This is the block version (OpenBLAS level 3) of TREVC. Faster but not so accurate.

Computing for type matrix<double>

```
bool  matrix::EigenVectorsTriangularZBlocked(
   ENUM_EIG_VECTORS      side,                  // compute left and/or right eigenve
   matrix&               schur_z,               // orthogonal matrix of Schur vecto
   matrix&               left_eigenvectors,     // matrix of computed left vectors
   matrix&               right_eigenvectors     // matrix of computed right vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenVectorsTriangularZBlocked(
   ENUM_EIG_VECTORS      side,                  // compute left and/or right eigenve
   matrixf&              schur_z,               // orthogonal matrix of Schur vecto
   matrixf&              left_eigenvectors,     // matrix of computed left vectors
   matrixf&              right_eigenvectors     // matrix of computed right vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenVectorsTriangularZBlocked(
   ENUM_EIG_VECTORS      side,                  // compute left and/or right eigenve
   matrixc&              schur_z,               // orthogonal matrix of Schur vecto
   matrixc&              left_eigenvectors,     // matrix of computed left vectors
   matrixc&              right_eigenvectors     // matrix of computed right vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenVectorsTriangularZBlocked(
   ENUM_EIG_VECTORS      side,                  // compute left and/or right eigenve
   matrixcf&             schur_z,               // orthogonal matrix of Schur vecto
   matrixcf&             left_eigenvectors,     // matrix of computed left vectors
   matrixcf&             right_eigenvectors     // matrix of computed right vectors
   );
```

Parameters

side

[in] ENUM_EIG_VECTORS enumeration value which determines what the eigenvectors are computed left, right or both.

© 2000–2025, MetaQuotes Ltd.

shur_z

[in] Orthogonal matrix of Schur vectors computed by EigenHessenbergSchurQ or EigenSolverSchur. Can be of zero size. In this case no backtransformation is produced.

left_eigenvectors

[out] Matrix of left eigenvectors.

right_eigenvectors

[out] Matrix of right eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The right eigenvector x and the left eigenvector y of T corresponding to an eigenvalue w are defined by:

T*x = w*x, (y**H)*T = w*(y**H)

where y**H denotes the conjugate transpose of y.

The eigenvalues are not input to this routine, but are read directly from the diagonal blocks of T.

This routine returns the matrices X and/or Y of right and left eigenvectors of T, or the products Q*X and/or Q*Y, where Q is an input matrix. If Q is the orthogonal factor that reduces a matrix A to Schur form T, then Q*X and Q*Y are the matrices of right and left eigenvectors of A.

ENUM_EIG_VECTORS

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVECTORS_N	Cannot be used with this method
EIGVECTORS_L	Only left eigenvectors are computed.
EIGVECTORS_R	Only right eigenvectors are computed.
EIGVECTORS_LR	Left and right eigenvectors are computed, eigenvalues are always computed.

© 2000–2025, MetaQuotes Ltd.

Symmetric Matrices

Functions for computing eigenvalues and eigenvectors of symmetric or Hermitian matrices using the divide and conquer algorithm, making the process efficient and fast. These methods can be applied to matrices of different data types, including real and complex numbers.

Function	Action
EigenSymmetricDC	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the divide-and-conquer algorithm (LAPACK functions SYEVD, HEEVD).
EigenSymmetricQR	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the QR algorithm (LAPACK functions SYEV, HEEV).
EigenSymmetricRobust	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK functions SYEVR, HEEVR).
EigenSymmetricBisect	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugate) matrix using the bisection algorithm (LAPACK functions SYEVX, HEEVX).
EigenSymmetricDC2s	Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal. If eigenvectors are desired, it uses a divide and conquer algorithm (LAPACK functions SYEVD_2STAGE, HEEVD_2STAGE).
EigenSymmetricQR2s	Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal (LAPACK functions SYEV_2STAGE, HEEV_2STAGE).
EigenSymmetricRobust2s	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal then using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK functions SYEVR_2STAGE, HEEVR_2STAGE).
EigenSymmetricBisect2s	Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction

© 2000–2025, MetaQuotes Ltd.

Function	Action
	to tridiagonal then using the bisection algorithm (LAPACK functions SYEVX_2STAGE, HEEVX_2STAGE).
EigenSymmetric2DC	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x. LAPACK functions SYGVD, HEGVD.
EigenSymmetric2QR	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x. LAPACK functions SYGV, HEGV.
EigenSymmetric2Bisect	Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x. LAPACK functions SYGVX, HEGVX.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricDC

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the divide-and-conquer algorithm (LAPACK functions SYEVD, HEEVD).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricDC(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricQR

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the QR algorithm (LAPACK functions SYEV, HEEV).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricQR(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricRobust

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK functions SYEVR, HEEVR).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // Upper bound of the subset
   float                 abstol,            // absolute error tolerance
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorf&             eigen_values,       // vector of computed eigenvectors
   matrixcf&            eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

© 2000–2025, MetaQuotes Ltd.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricBisect

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the bisection algorithm (LAPACK functions SYEVX, HEEVX).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // Upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

V

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

© 2000–2025, MetaQuotes Ltd.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricDC2s

Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal. If eigenvectors are desired, it uses a divide and conquer algorithm (LAPACK functions SYEVD_2STAGE, HEEVD_2STAGE).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricDC2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated. In the current OpenBLAS implementation, this value is not supported. Attempting to use it will result in error 4003 (ERR_INVALID_PARAMETER).

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated. Not available in this release.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricQR2s

Compute all eigenvalues and, optionally, eigenvectors of a real symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal (LAPACK functions SYEV_2STAGE, HEEV_2STAGE).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vector&               eigen_values,      // vector of computed eigenvalues
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricQR2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   vectorf&              eigen_values,      // vector of computed eigenvalues
   matrixcf&             eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated. In the current OpenBLAS implementation, this value is not supported. Attempting to use it will result in error 4003 (ERR_INVALID_PARAMETER).

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated. Not available in this release.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricRobust2s

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal then using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK functions SYEVR_2STAGE, HEEVR_2STAGE).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricRobust2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorf&              eigen_values,       // vector of computed eigenvectors
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated. In the current OpenBLAS implementation, this value is not supported. Attempting to use it will result in error 4003 (ERR_INVALID_PARAMETER).

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

© 2000–2025, MetaQuotes Ltd.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated. Not available in this release.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetricBisect2s

Compute eigenvalues and eigenvectors of a symmetric or Hermitian (complex conjugated) matrix using the 2stage technique for the reduction to tridiagonal then using the bisection algorithm (LAPACK functions SYEVX_2STAGE, HEEVX_2STAGE).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrixc&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetricBisect2s(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorf&             eigen_values,       // vector of computed eigenvectors
   matrixcf&            eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

V

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated. In the current OpenBLAS implementation, this value is not supported. Attempting to use it will result in error 4003 (ERR_INVALID_PARAMETER).

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

© 2000–2025, MetaQuotes Ltd.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated. Not available in this release.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetric2DC

Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form

A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

Here A and B are assumed to be symmetric (Hermitian) and B is also positive definite. If eigenvectors are desired, it uses a divide-and-conquer algorithm (LAPACK functions SYGVD, HEGVD).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrix&               B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors,      // matrix of computed eigenvectors
   matrix&               triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixf&              B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors,      // matrix of computed eigenvectors
   matrixf&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixc&              B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors,      // matrix of computed eigenvectors
   matrixc&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetric2DC(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixcf&             B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors,      // matrix of computed eigenvectors
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixcf&             triangular_factor   // triangular factor from the Cholesky fa
   );
```

Parameters

itype

[in] ENUM_EIGS2_TYPE enumeration value which specified the problem type to be solved : A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

B

[in] Second matrix B. Must be positive definite symmetric (or Hermitian conjugated) matrix.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

triangular_factor

[out] The triangular factor U or L from the Cholesky factorization B = U**T*U or B = L*L**T.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated). Second matrix B must be positive definite symmetric. If the input matrix and second matrix B are triangular, then both must be the same triangular, upper or lower.

ENUM_EIGS2_TYPE

An enumeration that specifies the problem type to be solved.

ID	Description
EIGS2TYPE_1	1: A*x = (lambda)*B*x
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetric2QR

Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form

A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

Here A and B are assumed to be symmetric (Hermitian) and B is also positive definite. This method uses QR algorithm (LAPACK functions SYGV, HEGV).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrix&               B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors,      // matrix of computed eigenvectors
   matrix&               triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixf&              B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors,      // matrix of computed eigenvectors
   matrixf&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixc&              B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors,      // matrix of computed eigenvectors
   matrixc&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetric2QR(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   matrixcf&             B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors,      // matrix of computed eigenvectors
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixcf&             triangular_factor   // triangular factor from the Cholesky fa
   );
```

Parameters

itype

[in] ENUM_EIGS2_TYPE enumeration value which specified the problem type to be solved : A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

B

[in] Second matrix B. Must be positive definite symmetric (or Hermitian conjugated) matrix.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

triangular_factor

[out] The triangular factor U or L from the Cholesky factorization B = U**T*U or B = L*L**T.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated). Second matrix B must be positive definite symmetric. If the input matrix and second matrix B are triangular, then both must be the same triangular, upper or lower.

ENUM_EIGS2_TYPE

An enumeration that specifies the problem type to be solved.

ID	Description
EIGS2TYPE_1	1: A*x = (lambda)*B*x
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

© 2000-2025, MetaQuotes Ltd.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenSymmetric2Bisect

Compute all the eigenvalues, and optionally, the eigenvectors of a generalized symmetric-definite eigenproblem, of the form

A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

Here A and B are assumed to be symmetric (Hermitian) and B is also positive definite. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues. This method uses bisection algorithm (LAPACK functions SYGVX, HEGVX).

Computing for type matrix<double>

```
bool  matrix::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,              // subset of eigenvalues to compute
   double                lower,              // lower bound of the subset
   double                upper,              // Upper bound of the subset
   double                abstol,             // absolute error tolerance
   matrix&               B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors,      // matrix of computed eigenvectors
   matrix&               triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,              // subset of eigenvalues to compute
   float                 lower,              // lower bound of the subset
   float                 upper,              // upper bound of the subset
   float                 abstol,             // absolute error tolerance
   matrixf&              B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors,      // matrix of computed eigenvectors
   matrixf&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,              // subset of eigenvalues to compute
   double                lower,              // lower bound of the subset
   double                upper,              // Upper bound of the subset
   double                abstol,             // absolute error tolerance
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&              B,                  // second matrix
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors,      // matrix of computed eigenvectors
   matrixc&              triangular_factor   // triangular factor from the Cholesky fa
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenSymmetric2Bisect(
   ENUM_EIGS2_TYPE       itype,              // the problem type to be solved
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,              // subset of eigenvalues to compute
   float                 lower,              // lower bound of the subset
   float                 upper,              // Upper bound of the subset
   float                 abstol,             // absolute error tolerance
   matrixcf&             B,                  // second matrix
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors,      // matrix of computed eigenvectors
   matrixcf&             triangular_factor   // triangular factor from the Cholesky fa
   );
```

Parameters

itype

[in] ENUM_EIGS2_TYPE enumeration value which specified the problem type to be solved : A*x=(lambda)*B*x, A*Bx=(lambda)*x, or B*A*x=(lambda)*x.

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

© 2000-2025, MetaQuotes Ltd.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

B

[in] Second matrix B. Must be positive definite symmetric (or Hermitian conjugated) matrix.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

triangular_factor

[out] The triangular factor U or L from the Cholesky factorization B = U**T*U or B = L*L**T.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated). Second matrix B must be positive definite symmetric. If the input matrix and second matrix B are triangular, then both must be the same triangular, upper or lower.

ENUM_EIGS2_TYPE

An enumeration that specifies the problem type to be solved.

ID	Description
EIGS2TYPE_1	1: A*x = (lambda)*B*x
EIGS2TYPE_2	2: A*B*x = (lambda)*x
EIGS2TYPE_3	3: B*A*x = (lambda)*x

© 2000-2025, MetaQuotes Ltd.

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

Tridiagonal Matrices

Functions for computing eigenvalues and eigenvectors of symmetric tridiagonal matrices using various algorithms. Each function implements a specific solution method and supports matrix types double and float.

Common Parameters:

- jobv — Determines whether to compute eigenvectors (EIGVALUES_V) or only eigenvalues (EIGVALUES_N).
- range — Specifies the range of computed eigenvalues (BLASRANGE_A, BLASRANGE_V, BLASRANGE_I).
- lower and upper — Lower and upper bounds for computing a subset of the spectrum.
- abstol — Absolute error tolerance.

All functions operate on symmetric tridiagonal matrices and allow selecting the most suitable algorithm depending on performance and accuracy requirements.

Function	Action
EigenTridiagonalDC	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (LAPACK function STEVD).
EigenTridiagonalQR	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (LAPACK function STEV).
EigenTridiagonalRobust	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK function STEVR).
EigenTridiagonalBisect	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the bisection algorithm (LAPACK function STEVX).
EigenTridiagonalQL	Compute all eigenvalues of a symmetric tridiagonal matrix using the Pal-Walker-Kahan variant of the QL or QR algorithm (LAPACK function STERF).
EigenTridiagonalDCQ	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (LAPACK function STEDC).
EigenTridiagonalQRQ	Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (LAPACK function STEQR).
EigenTridiagonalPosDefQ	Compute eigenvalues and eigenvectors of a symmetric positive definite (положительно

© 2000–2025, MetaQuotes Ltd.

Function	Action
	определённая) tridiagonal matrix using the QR algorithm (LAPACK function PTEQR).

© 2000–2025, MetaQuotes Ltd.

EigenTridiagonalDC

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (LAPACK function STEVD).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalDC(
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalDC(
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

© 2000–2025, MetaQuotes Ltd.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenTridiagonalQR

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (LAPACK function STEV).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalQR(
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalQR(
   ENUM_EIG_VALUES       jobv,               // compute eigenvectors or not
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the jobv parameter.

When jobv = EIGVALUES_V, eigenvectors and eigenvalues are calculated.

If EIGVALUES_N is set, eigenvectors are not calculated. Only eigenvalues are computed.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIG_VALUES

An enumeration that specifies whether to calculate eigenvectors.

© 2000–2025, MetaQuotes Ltd.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are calculated, without vectors.

© 2000–2025, MetaQuotes Ltd.

EigenTridiagonalRobust

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the Multiple Relatively Robust Representations, MRRR algorithm (LAPACK function STEVR).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalRobust(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

© 2000–2025, MetaQuotes Ltd.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

EigenTridiagonalBisect

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the bisection algorithm (LAPACK function STEVX).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   double                lower,             // lower bound of the subset
   double                upper,             // Upper bound of the subset
   double                abstol,            // absolute error tolerance
   vector&               eigen_values,      // vector of computed eigenvectors
   matrix&               eigen_vectors      // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalBisect(
   ENUM_EIG_VALUES       jobv,              // compute eigenvectors or not
   ENUM_BLAS_RANGE       range,             // subset of eigenvalues to compute
   float                 lower,             // lower bound of the subset
   float                 upper,             // upper bound of the subset
   float                 abstol,            // absolute error tolerance
   vectorf&              eigen_values,      // vector of computed eigenvectors
   matrixf&              eigen_vectors      // matrix of computed eigenvectors
   );
```

Parameters

jobv

[in] ENUM_EIG_VALUES enumeration value which determines the method for computing eigenvectors.

range

[in] ENUM_BLAS_RANGE enumeration value that defines a subset of computable eigenvalues and vectors.

lower

[in] The lower bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

upper

[in] The upper bound of eigenvalues subset; it is specified depending on the value of the '*range*' parameter.

abstol

[in] Absolute error tolerance.

The absolute error tolerance to which each eigenvalue/eigenvector is required.

© 2000–2025, MetaQuotes Ltd.

If jobv = 'V', the eigenvalues and eigenvectors output have residual norms bounded by abstol, and the dot products between different eigenvectors are bounded by abstol.

If abstol < n *eps*|T|, then n *eps*|T| is used instead, where eps is the machine precision, and |T| is the 1-norm of the matrix T. The eigenvalues are computed to an accuracy of eps*|T| irrespective of abstol.

If high relative precision is important, 'abstol' should be set to a safe minimum value X such that 1.0/X does not overflow.

eigen_values

[out] Vector of eigenvalues.

V

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the values of the *jobv* and *range* parameters.

When BLASRANGE_A is set, all eigenvalues are computed, and the lower and upper parameters are ignored.

With the BLASRANGE_V value, only those eigenvalues (and their vectors) are computed, which fall within the range of real values specified by the 'lower' and 'upper' parameters.

With the BLASRANGE_I value, only those eigenvalues (and their vectors) are computed, which fall within the range of integer indices specified by the 'lower' and 'upper' parameters. For example, with lower=0 and upper=2, only the first three eigenvalues are computed.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIG_VALUES

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGVALUES_V	Eigenvectors and eigenvalues are calculated.
EIGVALUES_N	Only eigenvalues are computed, without vectors.

ENUM_BLAS_RANGE

An enumeration defining how right singular vectors should be computed.

ID	Description
BLASRANGE_A	All singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASRANGE_V	All singular or eigenvalues in the half-open interval (VL,VU] will be found.
BLASRANGE_I	The IL-th through IU-th singular or eigenvalues will be found.

© 2000–2025, MetaQuotes Ltd.

EigenTridiagonalQL

Compute all eigenvalues of a symmetric tridiagonal matrix using the Pal-Walker-Kahan variant of the QL or QR algorithm (LAPACK function STERF).

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalQL(
   vector&               eigen_values        // vector of computed eigenvalues
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalQL(
   vectorf&              eigen_values        // vector of computed eigenvalues
   );
```

Parameters

eigen_values

[out] Vector of eigenvalues.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input must be a symmetric matrix in the tridiagonal form.

© 2000–2025, MetaQuotes Ltd.

EigenTridiagonalDCQ

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the divide-and-conquer algorithm (LAPACK function STEDC). Unlike EigenTridiagonalDC, this method can be used to compute the eigenvectors of the original symmetric matrix. A symmetric matrix can be reduced to tridiagonal form using the ReduceSymmetricToTridiagonal method. The orthogonal matrix Q obtained from this transformation is then used to compute the eigenvectors of the original symmetric matrix.

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrix&               Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixf&              Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixc&              Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenTridiagonalDCQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixcf&             Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

compv

[in] ENUM_EIGTRIDIAG_Z enumeration value which determines the method for computing eigenvectors.

Q

[in] Orthogonal matrix Q produced by method ReflectTridiagonalToQ.

© 2000–2025, MetaQuotes Ltd.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the compv parameter.

When compv = EIGCOMPZ_N, compute eigenvalues only, eigenvectors are not calculated.

If EIGCOMPZ_V is set, eigenvalues are computed and eigenvectors of original symmetric matrix are calculated also.

If EIGCOMPZ_I is set, eigenvalues are computed and eigenvectors of tridiagonal matrix are calculated also.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIGTRIDIAG_Z

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGCOMPZ_N	'N': Compute eigenvalues only
EIGCOMPZ_V	'V': Compute eigenvectors of original symmetric matrix also
EIGCOMPZ_I	'I': Compute eigenvectors of tridiagonal matrix also

© 2000–2025, MetaQuotes Ltd.

EigenTridiagonalQRQ

Compute eigenvalues and eigenvectors of a symmetric tridiagonal matrix using the QR algorithm (LAPACK function STEQR). Unlike EigenTridiagonalQR, this method can be used to compute the eigenvectors of the original symmetric matrix. A symmetric matrix can be reduced to tridiagonal form using the ReduceSymmetricToTridiagonal method. The orthogonal matrix Q obtained from this transformation is then used to compute the eigenvectors of the original symmetric matrix.

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrix&               Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixf&              Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixc&              Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenTridiagonalQRQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixcf&             Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

compv

[in] ENUM_EIGTRIDIAG_Z enumeration value which determines the method for computing eigenvectors.

Q

[in] Orthogonal matrix Q produced by method ReflectTridiagonalToQ.

© 2000–2025, MetaQuotes Ltd.

eigen_values

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the compv parameter.

When compv = EIGCOMPZ_N, compute eigenvalues only, eigenvectors are not calculated.

If EIGCOMPZ_V is set, eigenvalues are computed and eigenvectors of original symmetric matrix are calculated also.

If EIGCOMPZ_I is set, eigenvalues are computed and eigenvectors of tridiagonal matrix are calculated also.

The input must be a symmetric matrix in the tridiagonal form.

ENUM_EIGTRIDIAG_Z

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGCOMPZ_N	'N': Compute eigenvalues only
EIGCOMPZ_V	'V': Compute eigenvectors of original symmetric matrix also
EIGCOMPZ_I	'I': Compute eigenvectors of tridiagonal matrix also

© 2000–2025, MetaQuotes Ltd.

EigenTridiagonalPosDefQ

Compute eigenvalues and eigenvectors of a symmetric positive definite tridiagonal matrix using the QR algorithm (LAPACK function PTEQR). This method can be used to compute the eigenvectors of both the input tridiagonal matrix and the original positive definite symmetric matrix. This is achieved using the orthogonal matrix Q obtained during the reduction to tridiagonal form.

Computing for type matrix<double>

```
bool  matrix::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrix&               Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrix&               eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixf&              Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixf&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixc&              Q,                  // orthogonal matrix used in the reductio
   vector&               eigen_values,       // vector of computed eigenvalues
   matrixc&              eigen_vectors       // matrix of computed eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenTridiagonalPosDefQ(
   ENUM_EIGTRIDIAG_Z     compv,              // compute eigenvectors or not
   matrixcf&             Q,                  // orthogonal matrix used in the reductio
   vectorf&              eigen_values,       // vector of computed eigenvalues
   matrixcf&             eigen_vectors       // matrix of computed eigenvectors
   );
```

Parameters

compv

[in] ENUM_EIGTRIDIAG_Z enumeration value which determines the method for computing eigenvectors.

Q

[in] Orthogonal matrix Q produced by method ReflectTridiagonalToQ.

eigen_values

© 2000–2025, MetaQuotes Ltd.

[out] Vector of eigenvalues.

eigen_vectors

[out] Matrix of eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Computation depends on the value of the compv parameter.

When compv = EIGCOMPZ_N, compute eigenvalues only, eigenvectors are not calculated.

If EIGCOMPZ_V is set, eigenvalues are computed and eigenvectors of original symmetric matrix are calculated also.

If EIGCOMPZ_I is set, eigenvalues are computed and eigenvectors of tridiagonal matrix are calculated also.

The input must be a symmetric positive definite tridiagonal matrix.

ENUM_EIGTRIDIAG_Z

An enumeration that specifies whether to calculate eigenvectors.

ID	Description
EIGCOMPZ_N	'N': Compute eigenvalues only
EIGCOMPZ_V	'V': Compute eigenvectors of original symmetric matrix also
EIGCOMPZ_I	'I': Compute eigenvectors of tridiagonal matrix also

© 2000–2025, MetaQuotes Ltd.

Linear Equations

This section provides functions for solving systems of linear equations of the form A·X = B, where A is a square coefficient matrix, B is the right-hand side, and X is the solution matrix or vector. The methods support various data types (double, float, complex, complexf) and are based on LAPACK functions to ensure high precision and performance.

The functions are categorized based on the type of the coefficient matrix A. Each function is overloaded to support both matrix and vector forms for the right-hand side (B) and the solution (X).

Function	Action
LinearEquationsSolution	Computes the solution to the system of linear equations with a square coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function GESV.
LinearEquationsSolutionTriangular	Computes the solution to the system of linear equations with a square triangular coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function TRTRS.
InverseTriangular	Computes the inverse of a upper or lower triangular matrix A. LAPACK function TRTRI.
CondNumReciprocalTriangular	Estimates the reciprocal of the condition number of a upper or lower triangular matrix A in either the one-norm or infinity-norm. LAPACK function TRCON.
LinearEquationsSolutionSy	Computes the solution to the system of linear equations with a symmetric or Hermitian conjugated matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric or Hermitian conjugated matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK functions SYSV, HESV.
LinearEquationsSolutionComplexSy	Computes the solution to the system of linear equations with a complex symmetric (not Hermitian conjugated!) matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n complex symmetric matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function SYSV.

© 2000–2025, MetaQuotes Ltd.

Function	Action
LinearEquationsSolutionSyPD	Computes the solution to the system of linear equations with a symmetric or Hermitian conjugated positive-definite matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric or Hermitian conjugated positive-definite matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function POSV.
LinearEquationsSolutionGeTrid	Computes the solution to the system of linear equations with a general (nonsymmetric) tridiagonal coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general tridiagonal matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function GTSV.
LinearEquationsSolutionSyTridPD	Computes the solution to the system of linear equations with a symmetric tridiagonal positive-definite coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric tridiagonal matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function PTSV.
SylvesterEquationTriangular	Solves Sylvester equation for real quasi-triangular or complex triangular matrices: op(A)*X + X*op(B) = scale*C or op(A)*X - X*op(B) = scale*C where op(A) = A or A**T or A**H, and A and B are both upper triangular. LAPACK function TRSYL.
SylvesterEquationTriangularBlocked	Solves Sylvester equation for real quasi-triangular or complex triangular matrices: op(A)*X + X*op(B) = scale*C or op(A)*X - X*op(B) = scale*C where op(A) = A or A**T or A**H, and A and B are both upper triangular. LAPACK function TRSYL3. This is the block (BLAS level 3) version of TRSYL. Faster up to 5 times but not so accurate.

© 2000–2025, MetaQuotes Ltd.

LinearEquationsSolution

Computes the solution to the system of linear equations with a square coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function GESV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolution(
   matrix&          B,             // right hand side matrix B
   matrix&          X              // solution matrix X
   );

bool  matrix::LinearEquationsSolution(
   vector&          B,             // right hand side vector B
   vector&          X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolution(
   matrixf&         B,             // right hand side matrix B
   matrixf&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolution(
   vectorf&          B,             // right hand side vector B
   vectorf&          X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolution(
   matrixc&         B,             // right hand side matrix B
   matrixc&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolution(
   vectorc&         B,             // right hand side vector B
   vectorc&         X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolution(
   matrixcf&        B,             // right hand side matrix B
   matrixcf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolution(
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // right hand side vector B
   vectorcf&       X               // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

LinearEquationsSolutionTriangular

Computes the solution to the system of linear equations with a square triangular coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function TRTRS.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrix&             B,            // right hand side matrix B
   matrix&             X             // solution matrix X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vector&             B,            // right hand side vector B
   vector&             X             // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixf&            B,            // right hand side matrix B
   matrixf&            X             // solution matrix X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorf&            B,            // right hand side vector B
   vectorf&            X             // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixc&            B,            // right hand side matrix B
   matrixc&            X             // solution matrix X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorc&            B,            // right hand side vector B
   vectorc&            X             // solution vector X
   );
```

Computing for type matrix<complexf>

© 2000–2025, MetaQuotes Ltd.

```
bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixcf&           B,            // right hand side matrix B
   matrixcf&           X             // solution matrix X
   );

bool  matrix::LinearEquationsSolutionTriangular(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorcf&           B,            // right hand side vector B
   vectorcf&           X             // solution vector X
   );
```

Parameters

trans

[in] ENUM_EQUATIONS_FORM enumeration value which specifies the form of the system of equations.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

ENUM_EQUATIONS_FORM

An enumeration defining which form of the equations' system calculated.

ID	Description
EQUATIONSFORM_N	'N': A * X = B (No transpose)
EQUATIONSFORM_T	'T': A**T * X = B (Transpose)
EQUATIONSFORM_C	'C': A**H * X = B (Conjugate transpose)

In case of real matrices the value EQUATIONSFORM_C assumed as Transpose.

© 2000-2025, MetaQuotes Ltd.

InverseTriangular

Computes the inverse of a upper or lower triangular matrix A. LAPACK function TRTRI.

Computing for type matrix<double>

```
bool  matrix::InverseTriangular(
   matrix&          AI             // inverse of matrix A
   );
```

Computing for type matrix<float>

```
bool  matrixf::InverseTriangular(
   matrixf&         AI             // inverse of matrix A
   );
```

Computing for type matrix<complex>

```
bool  matrixc::InverseTriangular(
   matrixc&         AI             // inverse of matrix A
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::InverseTriangular(
   matrixcf&        AI             // inverse of matrix A
   );
```

Parameters

AI

[out] Inverted matrix.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

CondNumReciprocalTriangular

Estimates the reciprocal of the condition number of a upper or lower triangular matrix A in either the one-norm or infinity-norm. LAPACK function TRCON.

Computing for type matrix<double>

```
bool  matrix::CondNumReciprocalTriangular(
   ENUM_BLAS_NORM  norm           // matrix norm
   double&         rcond          // condition number reciprocal
   );
```

Computing for type matrix<float>

```
bool  matrixf::CondNumReciprocalTriangular(
   ENUM_BLAS_NORM  norm           // matrix norm
   float&          rcond          // condition number reciprocal
   );
```

Computing for type matrix<complex>

```
bool  matrixc::CondNumReciprocalTriangular(
   ENUM_BLAS_NORM  norm           // matrix norm
   double&         rcond          // condition number reciprocal
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::CondNumReciprocalTriangular(
   ENUM_BLAS_NORM  norm           // matrix norm
   float&          rcond          // condition number reciprocal
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORM enumeration, which defines estimation the reciprocal of the condition number.

rcond

[out] An estimate of the reciprocal of the condition number. The routine sets rcond=0 if the estimate underflows; in this case the matrix is singular (to working precision). However, anytime rcond is small compared to 1.0, for the working precision, the matrix may be poorly conditioned or even singular.

Return Value

Return true if successful, otherwise false in case of an error.

Note

© 2000–2025, MetaQuotes Ltd.

The computed *rcond* is never less than r (the reciprocal of the true condition number) and in practice is nearly always less than 10r. A call to this routine involves solving a number of systems of linear equations A*x = b; the number is usually 4 or 5 and never more than 11

ENUM_BLAS_NORM

An enumeration defining the matrix norm.

ID	Description
BLASNORM_O	'O': One-norm
BLASNORM_I	'I': Infinity-norm

© 2000–2025, MetaQuotes Ltd.

LinearEquationsSolutionSy

Computes the solution to the system of linear equations with a symmetric or Hermitian conjugated matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric or unitary matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK functions SYSV, HESV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionSy(
   matrix&          B,             // right hand side matrix B
   matrix&          X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSy(
   vector&          B,             // right hand side vector B
   vector&          X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixf&         B,             // right hand side matrix B
   matrixf&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSy(
   vectorf&          B,             // right hand side vector B
   vectorf&          X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixc&         B,             // right hand side matrix B
   matrixc&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSy(
   vectorc&         B,             // right hand side vector B
   vectorc&         X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSy(
   matrixcf&        B,             // right hand side matrix B
   matrixcf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSy(
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // right hand side vector B
   vectorcf&       X               // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

The input can be a symmetric (Hermitian conjugated), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000–2025, MetaQuotes Ltd.

LinearEquationsSolutionComplexSy

Computes the solution to the system of linear equations with a complex symmetric (not Hermitian conjugated!) matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n complex symmetric matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function SYSV.

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionComplexSy(
   matrixc&        B,             // right hand side matrix B
   matrixc&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionComplexSy(
   vectorc&        B,             // right hand side vector B
   vectorc&        X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionComplexSy(
   matrixcf&       B,             // right hand side matrix B
   matrixcf&       X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionComplexSy(
   vectorcf&       B,             // right hand side vector B
   vectorcf&       X              // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

The input can be a complex symmetric, upper triangular or lower triangular matrix. Triangular matrices are assumed to be complex symmetric.

© 2000–2025, MetaQuotes Ltd.

LinearEquationsSolutionSyPD

Computes the solution to the system of linear equations with a symmetric or Hermitian conjugated positive-definite matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric or unitary positive-definite matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function POSV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrix&          B,             // right hand side matrix B
   matrix&          X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vector&          B,             // right hand side vector B
   vector&          X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixf&         B,             // right hand side matrix B
   matrixf&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vectorf&          B,             // right hand side vector B
   vectorf&          X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixc&         B,             // right hand side matrix B
   matrixc&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyPD(
   vectorc&         B,             // right hand side vector B
   vectorc&         X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSyPD(
   matrixcf&        B,             // right hand side matrix B
   matrixcf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyPD(
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // right hand side vector B
   vectorcf&       X               // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

The input can be a symmetric (Hermitian conjugated), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000–2025, MetaQuotes Ltd.

LinearEquationsSolutionGeTrid

Computes the solution to the system of linear equations with a general (nonsymmetric) tridiagonal coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n general tridiagonal matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function GTSV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionGeTrid(
   matrix&          B,             // right hand side matrix B
   matrix&          X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionGeTrid(
   vector&          B,             // right hand side vector B
   vector&          X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionGeTrid(
   matrixf&         B,             // right hand side matrix B
   matrixf&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionGeTrid(
   vectorf&          B,             // right hand side vector B
   vectorf&          X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionGeTrid(
   matrixc&         B,             // right hand side matrix B
   matrixc&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionGeTrid(
   vectorc&         B,             // right hand side vector B
   vectorc&         X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionGeTrid(
   matrixcf&        B,             // right hand side matrix B
   matrixcf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionGeTrid(
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // right hand side vector B
   vectorcf&       X               // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

LinearEquationsSolutionSyTridPD

Computes the solution to the system of linear equations with a symmetric tridiagonal positive-definite coefficient matrix A and multiple right-hand sides. A*X = B, where A is an n-by-n symmetric tridiagonal matrix, the columns of matrix B are individual right-hand sides, and the columns of X are the corresponding solutions. LAPACK function PTSV.

Computing for type matrix<double>

```
bool  matrix::LinearEquationsSolutionSyTridPD(
   matrix&          B,             // right hand side matrix B
   matrix&          X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyTridPD(
   vector&          B,             // right hand side vector B
   vector&          X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrix::LinearEquationsSolutionSyTridPD(
   matrixf&         B,             // right hand side matrix B
   matrixf&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyTridPD(
   vectorf&          B,             // right hand side vector B
   vectorf&          X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrix::LinearEquationsSolutionSyTridPD(
   matrixc&         B,             // right hand side matrix B
   matrixc&         X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyTridPD(
   vectorc&         B,             // right hand side vector B
   vectorc&         X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::LinearEquationsSolutionSyTridPD(
   matrixcf&        B,             // right hand side matrix B
   matrixcf&        X              // solution matrix X
   );

bool  matrix::LinearEquationsSolutionSyTridPD(
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&       B,              // right hand side vector B
   vectorcf&       X               // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

SylvesterEquationTriangular

Solves Sylvester equation for real quasi-triangular or complex triangular matrices:

op(A)*X + X*op(B) = scale*C

or

op(A)*X - X*op(B) = scale*C

where op(A) = A or A**T or A**H, and A and B are both upper triangular. A is m-by-m and B is n-by-n; the right hand side C and the solution X are m-by-n; and scale is an output scale factor, set <= 1 to avoid overflow in X.

LAPACK function TRSYL.

Computing for type matrix<double>

```
bool  matrix::SylvesterEquationTriangular(
   ENUM_BLAS_TRANS       transa,        // specifies option op(A)
   ENUM_BLAS_TRANS       transb,        // specifies option op(B)
   ENUM_SYLVESTER_ISIGN  isign,         // specifies the sign in the equation
   matrix&               B,             // upper triangular matrix B
   matrix&               C,             // right hand side matrix C
   matrix&               X,             // solution matrix X
   double&               scale          // scale factor
   );
```

Computing for type matrix<float>

```
bool  matrixf::SylvesterEquationTriangular(
   ENUM_BLAS_TRANS       transa,        // specifies option op(A)
   ENUM_BLAS_TRANS       transb,        // specifies option op(B)
   ENUM_SYLVESTER_ISIGN  isign,         // specifies the sign in the equation
   matrixf&              B,             // upper triangular matrix B
   matrixf&              C,             // right hand side matrix C
   matrixf&              X,             // solution matrix X
   float&                scale          // scale factor
   );
```

Computing for type matrix<complex>

```
bool  matrix::SylvesterEquationTriangular(
   ENUM_BLAS_TRANS       transa,        // specifies option op(A)
   ENUM_BLAS_TRANS       transb,        // specifies option op(B)
   ENUM_SYLVESTER_ISIGN  isign,         // specifies the sign in the equation
   matrixc&              B,             // upper triangular matrix B
   matrixc&              C,             // right hand side matrix C
   matrixc&              X,             // solution matrix X
   double&               scale          // scale factor
   );
```

© 2000–2025, MetaQuotes Ltd.

Computing for type matrix<complexf>

```
bool  matrix::SylvesterEquationTriangular(
   ENUM_BLAS_TRANS       transa,        // specifies option op(A)
   ENUM_BLAS_TRANS       transb,        // specifies option op(B)
   ENUM_SYLVESTER_ISIGN  isign,         // specifies the sign in the equation
   matrixcf&             B,             // upper triangular matrix B
   matrixcf&             C,             // right hand side matrix C
   matrixcf&             X,             // solution matrix X
   float&                scale          // scale factor
   );
```

Parameters

transa

[in] ENUM_BLAS_TRANS enumeration value which specifies option op(A).

transb

[in] ENUM_BLAS_TRANS enumeration value which specifies option op(B).

isign

[in] ENUM_SYLVESTER_ISIGN enumeration value which specifies the sign in the equation.

B

[in] Upper triangular matrix B.

C

[in] Matrix C whose columns are the right-hand sides for the systems of equations.

X

[out] Matrix X with solution of Sylvester equation.

scale

[out] The scale factor, set <= 1 to avoid overflow in X.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix C.

ENUM_BLAS_TRANS

An enumeration defining options op(A) and op(B).

ID	Description
BLASTRANS_N	'N': No transpose

© 2000-2025, MetaQuotes Ltd.

ID	Description
BLASTRANS_T	'T': Transpose
BLASTRANS_C	'C': Conjugate transpose

In case of real matrices the value BLASTRANS_C assumed as Transpose.

In case of complex matrices the value BLASTRANS_T assumed as Conjugate transpose.

ENUM_SYLVESTER_ISIGN

An enumeration defining the sign in the equation.

ID	Description
ISIGN_PLUS	+1: solve op(A)*X + X*op(B) = scale*C
ISIGN_MINUS	-1: solve op(A)*X - X*op(B) = scale*C

© 2000–2025, MetaQuotes Ltd.

SylvesterEquationTriangularBlocked

Solves Sylvester equation for real quasi-triangular or complex triangular matrices:

op(A)*X + X*op(B) = scale*C

or

op(A)*X - X*op(B) = scale*C

where op(A) = A or A**T or A**H, and A and B are both upper triangular. A is m-by-m and B is n-by-n; the right hand side C and the solution X are m-by-n; and scale is an output scale factor, set <= 1 to avoid overflow in X.

LAPACK function TRSYL3. This is the block (BLAS level 3) version of TRSYL. Faster up to 5 times but not so accurate.

Computing for type matrix<double>

```
bool  matrix::SylvesterEquationTriangularBlocked(
   ENUM_BLAS_TRANS      transa,       // specifies option op(A)
   ENUM_BLAS_TRANS      transb,       // specifies option op(B)
   ENUM_SYLVESTER_ISIGN isign,        // specifies the sign in the equation
   matrix&              B,            // upper triangular matrix B
   matrix&              C,            // right hand side matrix C
   matrix&              X,            // solution matrix X
   double&              scale         // scale factor
   );
```

Computing for type matrix<float>

```
bool  matrixf::SylvesterEquationTriangularBlocked(
   ENUM_BLAS_TRANS      transa,       // specifies option op(A)
   ENUM_BLAS_TRANS      transb,       // specifies option op(B)
   ENUM_SYLVESTER_ISIGN isign,        // specifies the sign in the equation
   matrixf&             B,            // upper triangular matrix B
   matrixf&             C,            // right hand side matrix C
   matrixf&             X,            // solution matrix X
   float&               scale         // scale factor
   );
```

Computing for type matrix<complex>

```
bool  matrix::SylvesterEquationTriangularBlocked(
   ENUM_BLAS_TRANS      transa,       // specifies option op(A)
   ENUM_BLAS_TRANS      transb,       // specifies option op(B)
   ENUM_SYLVESTER_ISIGN isign,        // specifies the sign in the equation
   matrixc&             B,            // upper triangular matrix B
   matrixc&             C,            // right hand side matrix C
   matrixc&             X,            // solution matrix X
   double&              scale         // scale factor
```

© 2000–2025, MetaQuotes Ltd.

```
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SylvesterEquationTriangularBlocked(
   ENUM_BLAS_TRANS      transa,       // specifies option op(A)
   ENUM_BLAS_TRANS      transb,       // specifies option op(B)
   ENUM_SYLVESTER_ISIGN isign,        // specifies the sign in the equation
   matrixcf&            B,            // upper triangular matrix B
   matrixcf&            C,            // right hand side matrix C
   matrixcf&            X,            // solution matrix X
   float&               scale         // scale factor
   );
```

Parameters

transa

[in] ENUM_BLAS_TRANS enumeration value which specifies option op(A).

transb

[in] ENUM_BLAS_TRANS enumeration value which specifies option op(B).

isign

[in] ENUM_SYLVESTER_ISIGN enumeration value which specifies the sign in the equation.

B

[in] Upper triangular matrix B.

C

[in] Matrix C whose columns are the right-hand sides for the systems of equations.

X

[out] Matrix X with solution of Sylvester equation.

scale

[out] The scale factor, set <= 1 to avoid overflow in X.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

Least Squares

The Least Squares section provides a set of versatile functions for solving overdetermined or underdetermined linear systems of the form A * X ≈ B using least squares methods. These functions support a variety of matrix types (float, double, complex, complexf) and implement several numerically stable algorithms based on standard LAPACK routines.

Supported Methods:

- QR / LQ factorization (LeastSquaresSolution) — the primary approach for solving systems with full-rank matrices.
- QR with compact WY representation (LeastSquaresSolutionWY) — an efficient method optimized for block operations.
- SVD-based methods (for rank-deficient matrices):
  - LeastSquaresSolutionDC — using a divide-and-conquer algorithm,
  - LeastSquaresSolutionSVD — using classical singular value decomposition.
- QR with column pivoting (QR with pivoting) (LeastSquaresSolutionQRPivot) — provides robust solutions for rank-deficient systems and computes the effective rank.
- Tall-Skinny QR / Short-Wide LQ (LeastSquaresSolutionQRTallSkinny) — optimized for very tall or wide matrices.

Key Features:

- Support for solving systems in various forms: with matrix A, its transpose (A^t), or Hermitian conjugate (A^H).
- Computation of residual vectors and singular values (for SVD-based methods).
- Estimation of the effective rank of matrix A.
- Support for both single right-hand side vectors and multiple right-hand side matrices.

Function	Action
LeastSquaresSolution	Solves overdetermined or underdetermined real / complex linear systems involving an m-by-n matrix A, or its transpose / conjugate-transpose, using a QR or LQ factorization of A. It is assumed that A has full rank. LAPACK function GELS.
LeastSquaresSolutionDC	Computes the minimum-norm solution to a real or complex linear least squares problem: minimize 2-norm(| b - A*x |) using the singular value decomposition (SVD) of A. A is an m-by-n matrix which may be rank-deficient. The divide and conquer algorithm makes very mild assumptions about floating point arithmetic. LAPACK function GELSD.
LeastSquaresSolutionSVD	Computes the minimum-norm solution to a real or complex linear least squares problem: minimize 2-norm(| b - A*x |) using the singular value

© 2000–2025, MetaQuotes Ltd.

Function	Action
	decomposition (SVD) of A. A is an m-by-n matrix which may be rank-deficient. LAPACK function GELSS.
LeastSquaresSolutionWY	Solves overdetermined or underdetermined real / complex linear systems involving an m-by-n matrix A, or its transpose / conjugate-transpose, using a QR or LQ factorization of A with compact WY representation of Q. It is assumed that A has full rank. LAPACK function GELST.
LeastSquaresSolutionsQRPivot	Computes the minimum-norm solution to a real or complex linear least squares problem: minimize 2-norm(| b - A*x |) using a complete orthogonal factorization of A. A is an m-by-n matrix which may be rank-deficient. LAPACK function GELSY.
LeastSquaresSolutionQRTallSkinny	Solves overdetermined or underdetermined real / complex linear systems involving an m-by-n matrix A, or its transpose / conjugate-transpose, using a tall skinny QR or short wide LQ factorization of A. It is assumed that A has full rank. LAPACK function GETSLS.

© 2000–2025, MetaQuotes Ltd.

LeastSquaresSolution

Solves overdetermined or underdetermined real / complex linear systems involving an m-by-n matrix A, or its transpose / conjugate-transpose, using a QR or LQ factorization of A. It is assumed that A has full rank.

The following options are provided:

1. If trans = 'N' and m >= n: find the least squares solution of an overdetermined system, i.e., solve the least squares problem `minimize || B - A*X ||`.

2. If trans = 'N' and m < n: find the minimum norm solution of an underdetermined system A * X = B.

3. If trans = 'T' and m >= n: find the minimum norm solution of an underdetermined system A**T * X = B.

4. If trans = 'T' and m < n: find the least squares solution of an overdetermined system, i.e., solve the least squares problem `minimize || B - A**T * X ||`.

LAPACK function GELS.

Computing for type matrix<double>

```
bool  matrix::LeastSquaresSolution(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrix&             B,            // right hand side matrix B
   matrix&             X,            // solution matrix X
   matrix&             residuals     // matrix with residual sums of squares
   );

bool  matrix::LeastSquaresSolution(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vector&             B,            // right hand side vector B
   vector&             X,            // solution vector X
   vector&             residuals     // vector with residual sums of squares
   );
```

Computing for type matrix<float>

```
bool  matrixf::LeastSquaresSolution(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixf&            B,            // right hand side matrix B
   matrixf&            X,            // solution matrix X
   matrixf&            residuals     // matrix with residual sums of squares
   );

bool  matrixf::LeastSquaresSolution(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorf&            B,            // right hand side vector B
   vectorf&            X,            // solution vector X
   vectorf&            residuals     // vector with residual sums of squares
```

© 2000-2025, MetaQuotes Ltd.

```
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LeastSquaresSolution(
   ENUM_EQUATIONS_FORM trans,         // form of the system of equations
   matrixc&            B,             // right hand side matrix B
   matrixc&            X,             // solution matrix X
   matrixc&            residuals      // matrix with residual sums of squares
   );

bool  matrixc::LeastSquaresSolution(
   ENUM_EQUATIONS_FORM trans,         // form of the system of equations
   vectorc&            B,             // right hand side vector B
   vectorc&            X,             // solution vector X
   vectorc&            residuals      // vector with residual sums of squares
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LeastSquaresSolution(
   ENUM_EQUATIONS_FORM trans,         // form of the system of equations
   matrixcf&           B,             // right hand side matrix B
   matrixcf&           X,             // solution matrix X
   matrixcf&           residuals      // matrix with residual sums of squares
   );

bool  matrixcf::LeastSquaresSolution(
   ENUM_EQUATIONS_FORM trans,         // form of the system of equations
   vectorcf&           B,             // right hand side vector B
   vectorcf&           X,             // solution vector X
   vectorcf&           residuals      // vector with residual sums of squares
   );
```

Parameters

trans

[in] ENUM_EQUATIONS_FORM enumeration value which specifies the form of the system of equations.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear least squares problem.

residuals

[out] Matrix or vector with the residual sum of squares for the solution in each column, given by the sum of squares of modulus of elements in that column. If trans='N' and m<=n or trans!='N' and m>=n, then *residuals* matrix (vector) is empty.

© 2000-2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

Note

If trans='N'. Matrix B must be of size m-by-nrhs, where nrhs is number of right-hand side vectors, matrix X is of size n-by-nrhs. If m>=n, matrix X contains nrhs least squares solution vectors. If m<n, matrix X contains nrhs minimum norm solution vectors.

If trans!='N'. Matrix B must be of size n-by-nrhs, matrix X is of size m-by-nrhs. If m>=n, matrix X contains nrhs minimum norm solution vectors. If m<n, matrix X contains nrhs least squares solution vectors.

ENUM_EQUATIONS_FORM

An enumeration defining which form of the equations' system calculated.

ID	Description
EQUATIONSFORM_N	'N': the linear system involves A (No transpose)
EQUATIONSFORM_T	'T': the linear system involves A**T (Transpose)
EQUATIONSFORM_C	'C': the linear system involves A**H (Conjugate transpose)

In case of real matrices the value EQUATIONSFORM_C assumed as Transpose.

In case of complex matrices the value EQUATIONSFORM_T assumed as Conjugate transpose.

© 2000–2025, MetaQuotes Ltd.

LeastSquaresSolutionDC

Computes the minimum-norm solution to a real or complex linear least squares problem `minimize 2-norm(| b - A*x |)` using the singular value decomposition (SVD) of A. A is an m-by-n matrix which may be rank-deficient. The divide and conquer algorithm makes very mild assumptions about floating point arithmetic. LAPACK function GELSD.

Computing for type matrix<double>

```
bool  matrix::LeastSquaresSolutionDC(
   matrix&          B,                // right hand side matrix B
   double           rcond,            // condition to determine matrix A rank
   matrix&          X,                // solution matrix X
   matrix&          residuals,        // matrix with residual sums of squares
   vector&          singular_values,  // vector of singular values
   long             rank              // effective rank of A
   );

bool  matrix::LeastSquaresSolutionDC(
   vector&          B,                // right hand side vector B
   double           rcond,            // condition to determine matrix A rank
   vector&          X,                // solution vector X
   vector&          residuals,        // vector with residual sums of squares
   vector&          singular_values,  // vector of singular values
   long             rank              // effective rank of A
   );
```

Computing for type matrix<float>

```
bool  matrixf::LeastSquaresSolutionDC(
   matrixf&         B,                // right hand side matrix B
   float            rcond,            // condition to determine matrix A rank
   matrixf&         X,                // solution matrix X
   matrixf&         residuals,        // matrix with residual sums of squares
   vectorf&         singular_values,  // vector of singular values
   long             rank              // effective rank of A
   );

bool  matrixf::LeastSquaresSolutionDC(
   vectorf&         B,                // right hand side vector B
   float            rcond,            // condition to determine matrix A rank
   vectorf&         X,                // solution vector X
   vectorf&         residuals,        // vector with residual sums of squares
   vectorf&         singular_values,  // vector of singular values
   long             rank              // effective rank of A
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LeastSquaresSolutionDC(
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixc&        B,                // right hand side matrix B
   double          rcond,            // condition to determine matrix A rank
   matrixc&        X,                // solution matrix X
   matrixc&        residuals,        // matrix with residual sums of squares
   vector&         singular_values,  // vector of singular values
   long            rank              // effective rank of A
   );

bool  matrixc::LeastSquaresSolutionDC(
   vectorc&        B,                // right hand side vector B
   double          rcond,            // condition to determine matrix A rank
   vectorc&        X,                // solution vector X
   vectorc&        residuals,        // vector with residual sums of squares
   vector&         singular_values,  // vector of singular values
   long            rank              // effective rank of A
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LeastSquaresSolutionDC(
   matrixcf&       B,                // right hand side matrix B
   float           rcond,            // condition to determine matrix A rank
   matrixcf&       X,                // solution matrix X
   matrixcf&       residuals,        // matrix with residual sums of squares
   vectorf&        singular_values,  // vector of singular values
   long            rank              // effective rank of A
   );

bool  matrixcf::LeastSquaresSolutionDC(
   vectorcf&       B,                // right hand side vector B
   float           rcond,            // condition to determine matrix A rank
   vectorcf&       X,                // solution vector X
   vectorcf&       residuals,        // vector with residual sums of squares
   vectorf&        singular_values,  // vector of singular values
   long            rank              // effective rank of A
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

rcond

[in] *rcond* is used to determine the effective rank of A. Singular values S(i) <= *rcond**S(1) are treated as zero. If *rcond* < 0, machine precision is used instead.

X

[out] Matrix or vector X with solutions of linear least squares problem.

© 2000-2025, MetaQuotes Ltd.

residuals

[out] Matrix or vector with the residual sum of squares for the solution in each column is given by the sum of squares of elements in that column. If m<=n, then residuals matrix(vector) is empty.

singular_values

[out] Vector of computed singular values.

rank

[out] The effective rank of A, i.e., the number of singular values which are greater than *rcond**S(1).

Return Value

Return true if successful, otherwise false in case of an error.

Note

Matrix B must be of size m-by-nrhs, where nrhs is number of right-hand side vectors, matrix X is of size n-by-nrhs.

The condition number of A in the 2-norm = S(1)/S(min(m,n)).

The problem is solved in three steps:

Step 1. Reduce the coefficient matrix A to bidiagonal form with Householder transformations, reducing the original problem into a "bidiagonal least squares problem" (BLS).

Step 2. Solve the BLS using a divide and conquer approach.

Step 3. Apply back all the Householder transformations to solve the original least squares problem.

© 2000–2025, MetaQuotes Ltd.

LeastSquaresSolutionSVD

Computes the minimum-norm solution to a real or complex linear least squares problem `minimize 2-norm(| b - A*x |)` using the singular value decomposition (SVD) of A. A is an m-by-n matrix which may be rank-deficient. LAPACK function GELSS.

Computing for type matrix<double>

```
bool  matrix::LeastSquaresSolutionSVD(
   matrix&          B,                // right hand side matrix B
   double           rcond,            // condition to determine matrix A rank
   matrix&          X,                // solution matrix X
   matrix&          residuals,        // matrix with residual sums of squares
   vector&          singular_values,  // vector of singular values
   long             rank              // effective rank of A
   );

bool  matrix::LeastSquaresSolutionSVD(
   vector&          B,                // right hand side vector B
   double           rcond,            // condition to determine matrix A rank
   vector&          X,                // solution vector X
   vector&          residuals,        // vector with residual sums of squares
   vector&          singular_values,  // vector of singular values
   long             rank              // effective rank of A
   );
```

Computing for type matrix<float>

```
bool  matrixf::LeastSquaresSolutionSVD(
   matrixf&         B,                // right hand side matrix B
   float            rcond,            // condition to determine matrix A rank
   matrixf&         X,                // solution matrix X
   matrixf&         residuals,        // matrix with residual sums of squares
   vectorf&         singular_values,  // vector of singular values
   long             rank              // effective rank of A
   );

bool  matrixf::LeastSquaresSolutionSVD(
   vectorf&         B,                // right hand side vector B
   float            rcond,            // condition to determine matrix A rank
   vectorf&         X,                // solution vector X
   vectorf&         residuals,        // vector with residual sums of squares
   vectorf&         singular_values,  // vector of singular values
   long             rank              // effective rank of A
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LeastSquaresSolutionSVD(
   matrixc&         B,                // right hand side matrix B
```

© 2000–2025, MetaQuotes Ltd.

```
   double          rcond,            // condition to determine matrix A rank
   matrixc&        X,                // solution matrix X
   matrixc&        residuals,        // matrix with residual sums of squares
   vector&         singular_values,  // vector of singular values
   long            rank              // effective rank of A
   );

bool  matrixc::LeastSquaresSolutionSVD(
   vectorc&        B,                // right hand side vector B
   double          rcond,            // condition to determine matrix A rank
   vectorc&        X,                // solution vector X
   vectorc&        residuals,        // vector with residual sums of squares
   vector&         singular_values,  // vector of singular values
   long            rank              // effective rank of A
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LeastSquaresSolutionSVD(
   matrixcf&       B,                // right hand side matrix B
   float           rcond,            // condition to determine matrix A rank
   matrixcf&       X,                // solution matrix X
   matrixcf&       residuals,        // matrix with residual sums of squares
   vectorf&        singular_values,  // vector of singular values
   long            rank              // effective rank of A
   );

bool  matrixcf::LeastSquaresSolutionSVD(
   vectorcf&       B,                // right hand side vector B
   float           rcond,            // condition to determine matrix A rank
   vectorcf&       X,                // solution vector X
   vectorcf&       residuals,        // vector with residual sums of squares
   vectorf&        singular_values,  // vector of singular values
   long            rank              // effective rank of A
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

rcond

[in] *rcond* is used to determine the effective rank of A. Singular values S(i) <= *rcond**S(1) are treated as zero. If *rcond* < 0, machine precision is used instead.

X

[out] Matrix or vector X with solutions of linear least squares problem.

residuals

© 2000–2025, MetaQuotes Ltd.

[out] Matrix or vector with the residual sum of squares for the solution in each column is given by the sum of squares of elements in that column. If m<=n, then residuals matrix(vector) is empty.

singular_values

[out] Vector of computed singular values.

rank

[out] The effective rank of A, i.e., the number of singular values which are greater than *rcond**S(1).

Return Value

Return true if successful, otherwise false in case of an error.

Note

Matrix B must be of size m-by-nrhs, where nrhs is number of right-hand side vectors, matrix X is of size n-by-nrhs.

The condition number of A in the 2-norm = S(1)/S(min(m,n)).

© 2000–2025, MetaQuotes Ltd.

LeastSquaresSolutionWY

Solves overdetermined or underdetermined real / complex linear systems involving an m-by-n matrix A, or its transpose / conjugate-transpose, using a QR or LQ factorization of A with compact WY representation of Q. It is assumed that A has full rank.

The following options are provided:

1. If trans = 'N' and m >= n: find the least squares solution of an overdetermined system, i.e., solve the least squares problem `minimize || B - A*X ||`.

2. If trans = 'N' and m < n: find the minimum norm solution of an underdetermined system A * X = B.

3. If trans = 'T' and m >= n: find the minimum norm solution of an underdetermined system A**T * X = B.

4. If trans = 'T' and m < n: find the least squares solution of an overdetermined system, i.e., solve the least squares problem `minimize || B - A**T * X ||`.

LAPACK function GELST.

Computing for type matrix<double>

```
bool  matrix::LeastSquaresSolutionWY(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrix&             B,            // right hand side matrix B
   matrix&             X,            // solution matrix X
   matrix&             residuals     // matrix with residual sums of squares
   );

bool  matrix::LeastSquaresSolutionWY(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vector&             B,            // right hand side vector B
   vector&             X,            // solution vector X
   vector&             residuals     // vector with residual sums of squares
   );
```

Computing for type matrix<float>

```
bool  matrixf::LeastSquaresSolutionWY(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixf&            B,            // right hand side matrix B
   matrixf&            X,            // solution matrix X
   matrixf&            residuals     // matrix with residual sums of squares
   );

bool  matrixf::LeastSquaresSolutionWY(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorf&            B,            // right hand side vector B
   vectorf&            X,            // solution vector X
   vectorf&            residuals     // vector with residual sums of squares
```

© 2000–2025, MetaQuotes Ltd.

```
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LeastSquaresSolutionWY(
   ENUM_EQUATIONS_FORM trans,         // form of the system of equations
   matrixc&            B,             // right hand side matrix B
   matrixc&            X,             // solution matrix X
   matrixc&            residuals      // matrix with residual sums of squares
   );

bool  matrixc::LeastSquaresSolutionWY(
   ENUM_EQUATIONS_FORM trans,         // form of the system of equations
   vectorc&            B,             // right hand side vector B
   vectorc&            X,             // solution vector X
   vectorc&            residuals      // vector with residual sums of squares
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LeastSquaresSolutionWY(
   ENUM_EQUATIONS_FORM trans,         // form of the system of equations
   matrixcf&           B,             // right hand side matrix B
   matrixcf&           X,             // solution matrix X
   matrixcf&           residuals      // matrix with residual sums of squares
   );

bool  matrixcf::LeastSquaresSolutionWY(
   ENUM_EQUATIONS_FORM trans,         // form of the system of equations
   vectorcf&           B,             // right hand side vector B
   vectorcf&           X,             // solution vector X
   vectorcf&           residuals      // vector with residual sums of squares
   );
```

Parameters

trans

[in] ENUM_EQUATIONS_FORM enumeration value which specifies the form of the system of equations.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear least squares problem.

residuals

[out] Matrix or vector with the residual sum of squares for the solution in each column is given by the sum of squares of modulus of elements in that column. If trans='N' and m<=n or trans!='N' and m>=n, then residuals matrix(vector) is empty.

© 2000-2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

Note

If trans='N'. Matrix B must be of size m-by-nrhs, where nrhs is number of right-hand side vectors, matrix X is of size n-by-nrhs. If m>=n, matrix X contains nrhs least squares solution vectors. If m<n, matrix X contains nrhs minimum norm solution vectors.

If trans!='N'. Matrix B must be of size n-by-nrhs, matrix X is of size m-by-nrhs. If m>=n, matrix X contains nrhs minimum norm solution vectors. If m<n, matrix X contains nrhs least squares solution vectors.

The abbreviation WY is not an official acronym derived from names or mathematical terms. Compact WY representation is a historical term that originates from the original research paper where this representation was first introduced:

Schreiber, R., & Van Loan, C. (1989). A Storage-Efficient WY Representation for Products of Householder Transformations. SIAM Journal on Scientific and Statistical Computing.

To construct the matrix Q in a QR or LQ factorization, Householder reflectors are used:

Q=H1H2…Hk,

where each Hi is a Householder transformation. In the classical implementation, each reflector is applied individually. In the block implementation, to accelerate computations and leverage the performance benefits of BLAS, the compact WY representation is used.

ENUM_EQUATIONS_FORM

An enumeration defining which form of the equations' system calculated.

ID	Description
EQUATIONSFORM_N	'N': the linear system involves A (No transpose)
EQUATIONSFORM_T	'T': the linear system involves A**T (Transpose)
EQUATIONSFORM_C	'C': the linear system involves A**H (Conjugate transpose)

In case of real matrices the value EQUATIONSFORM_C assumed as Transpose.

In case of complex matrices the value EQUATIONSFORM_T assumed as Conjugate transpose.

© 2000–2025, MetaQuotes Ltd.

LeastSquaresSolutionQRPivot

Computes the minimum-norm solution to a real or complex linear least squares problem `minimize 2-norm(| b - A*x |)` using a complete orthogonal factorization of A. A is an m-by-n matrix which may be rank-deficient. LAPACK function GELSY.

Computing for type matrix<double>

```
bool  matrix::LeastSquaresSolutionQRPivot(
   matrix&          B,                // right hand side matrix B
   long[]&          jpvt,             // array with predefined permutations
   double           rcond,            // condition to determine matrix A rank
   matrix&          X,                // solution matrix X
   long[]&          jpvt_out,         // array with proceeded permutations
   long             rank              // effective rank of A
   );

bool  matrix::LeastSquaresSolutionQRPivot(
   vector&          B,                // right hand side vector B
   long[]&          jpvt,             // array with predefined permutations
   double           rcond,            // condition to determine matrix A rank
   vector&          X,                // solution vector X
   long[]&          jpvt_out,         // array with proceeded permutations
   long             rank              // effective rank of A
   );
```

Computing for type matrix<float>

```
bool  matrixf::LeastSquaresSolutionQRPivot(
   matrixf&         B,                // right hand side matrix B
   long[]&          jpvt,             // array with predefined permutations
   float            rcond,            // condition to determine matrix A rank
   matrixf&         X,                // solution matrix X
   long[]&          jpvt_out,         // array with proceeded permutations
   long             rank              // effective rank of A
   );

bool  matrixf::LeastSquaresSolutionQRPivot(
   vectorf&         B,                // right hand side vector B
   long[]&          jpvt,             // array with predefined permutations
   float            rcond,            // condition to determine matrix A rank
   vectorf&         X,                // solution vector X
   long[]&          jpvt_out,         // array with proceeded permutations
   long             rank              // effective rank of A
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LeastSquaresSolutionQRPivot(
   matrixc&         B,                // right hand side matrix B
```

© 2000–2025, MetaQuotes Ltd.

```
   long[]&         jpvt,              // array with predefined permutations
   double          rcond,             // condition to determine matrix A rank
   matrixc&        X,                 // solution matrix X
   long[]&         jpvt_out,          // array with proceeded permutations
   long            rank               // effective rank of A
   );

bool  matrixc::LeastSquaresSolutionQRPivot(
   vectorc&        B,                 // right hand side vector B
   long[]&         jpvt,              // array with predefined permutations
   double          rcond,             // condition to determine matrix A rank
   vectorc&        X,                 // solution vector X
   long[]&         jpvt_out,          // array with proceeded permutations
   long            rank               // effective rank of A
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LeastSquaresSolutionQRPivot(
   matrixcf&       B,                 // right hand side matrix B
   long[]&         jpvt,              // array with predefined permutations
   float           rcond,             // condition to determine matrix A rank
   matrixcf&       X,                 // solution matrix X
   long[]&         jpvt_out,          // array with proceeded permutations
   long            rank               // effective rank of A
   );

bool  matrixcf::LeastSquaresSolutionQRPivot(
   vectorcf&       B,                 // right hand side vector B
   long[]&         jpvt,              // array with predefined permutations
   float           rcond,             // condition to determine matrix A rank
   vectorcf&       X,                 // solution vector X
   long[]&         jpvt_out,          // array with proceeded permutations
   long            rank               // effective rank of A
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

jpvt

[in] Integer array of dimension n. If jpvt(i) != 0, the i-th column of A is permuted to the front of AP, otherwise column i is a free column. If array has zero size (or not initialized), then all the columns of A assumed to be free.

rcond

© 2000–2025, MetaQuotes Ltd.

[in] *rcond* is used to determine the effective rank of A, which is defined as the order of the largest leading triangular submatrix R11 (see Note 2) in the QR factorization with pivoting of A, whose estimated condition number < 1/*rcond*.

X

[out] Matrix or vector X with solutions of linear least squares problem.

jpvt_out

[out] Integer array with proceeded permutations. If jpvt(i) = k, then the i-th column of AP was the k-th column of A.

rank

[out] The effective rank of A, that is, the order of the submatrix R11 (see Note 2). This is the same as the order of the submatrix T11 in the complete orthogonal factorization of A.

Return Value

Return true if successful, otherwise false in case of an error.

Note

1. Matrix B must be of size m-by-nrhs, where nrhs is number of right-hand side vectors, matrix X is of size n-by-nrhs.

2. The routine first computes a QR factorization with column pivoting:

```
A * P = Q * [ R11 R12 ]
            [  0  R22 ]
```

with R11 defined as the largest leading submatrix whose estimated condition number is less than 1/*rcond*. The order of R11, *rank*, is the effective rank of A.

Then, R22 is considered to be negligible, and R12 is annihilated by orthogonal transformations from the right, arriving at the complete orthogonal factorization:

```
A * P = Q * [ T11 0 ] * Z
            [  0  0 ]
```

The minimum-norm solution is then

```
X = P * Z**T [ inv(T11)*Q1**T*B ]
             [          0       ]
```

where Q1 consists of the first *rank* columns of Q.

© 2000–2025, MetaQuotes Ltd.

LeastSquaresSolutionQRTallSkinny

Solves overdetermined or underdetermined real / complex linear systems involving an m-by-n matrix A, or its transpose / conjugate-transpose, using a tall skinny QR or short wide LQ factorization of A. It is assumed that A has full rank.

The following options are provided:

1. If trans = 'N' and m >= n: find the least squares solution of an overdetermined system, i.e., solve the least squares problem `minimize || B - A*X ||`.

2. If trans = 'N' and m < n: find the minimum norm solution of an underdetermined system A * X = B.

3. If trans = 'T' and m >= n: find the minimum norm solution of an underdetermined system A**T * X = B.

4. If trans = 'T' and m < n: find the least squares solution of an overdetermined system, i.e., solve the least squares problem `minimize || B - A**T * X ||`.

LAPACK function GETSLS.

Computing for type matrix<double>

```
bool  matrix::LeastSquaresSolutionQRTallSkinny(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrix&             B,            // right hand side matrix B
   matrix&             X             // solution matrix X
   );

bool  matrix::LeastSquaresSolutionQRTallSkinny(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vector&             B,            // right hand side vector B
   vector&             X             // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrixf::LeastSquaresSolutionQRTallSkinny(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixf&            B,            // right hand side matrix B
   matrixf&            X             // solution matrix X
   );

bool  matrixf::LeastSquaresSolutionQRTallSkinny(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorf&            B,            // right hand side vector B
   vectorf&            X             // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LeastSquaresSolutionQRTallSkinny(
```

© 2000–2025, MetaQuotes Ltd.

```
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixc&            B,            // right hand side matrix B
   matrixc&            X             // solution matrix X
   );

bool  matrixc::LeastSquaresSolutionQRTallSkinny(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorc&            B,            // right hand side vector B
   vectorc&            X             // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LeastSquaresSolutionQRTallSkinny(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   matrixcf&           B,            // right hand side matrix B
   matrixcf&           X             // solution matrix X
   );

bool  matrixcf::LeastSquaresSolutionQRTallSkinny(
   ENUM_EQUATIONS_FORM trans,        // form of the system of equations
   vectorcf&           B,            // right hand side vector B
   vectorcf&           X             // solution vector X
   );
```

Parameters

trans

[in] ENUM_EQUATIONS_FORM enumeration value which specifies the form of the system of equations.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear least squares problem.

Return Value

Return true if successful, otherwise false in case of an error.

Note

If trans='N'. Matrix B must be of size m-by-nrhs, where nrhs is number of right-hand side vectors, matrix X is of size n-by-nrhs. If m>=n, matrix X contains nrhs least squares solution vectors. If m<n, matrix X contains nrhs minimum norm solution vectors.

If trans!='N'. Matrix B must be of size n-by-nrhs, matrix X is of size m-by-nrhs. If m>=n, matrix X contains nrhs minimum norm solution vectors. If m<n, matrix X contains nrhs least squares solution vectors.

© 2000–2025, MetaQuotes Ltd.

ENUM_EQUATIONS_FORM

An enumeration defining which form of the equations' system calculated.

ID	Description
EQUATIONSFORM_N	'N': the linear system involves A (No transpose)
EQUATIONSFORM_T	'T': the linear system involves A**T (Transpose)
EQUATIONSFORM_C	'C': the linear system involves A**H (Conjugate transpose)

In case of real matrices the value EQUATIONSFORM_C assumed as Transpose.

In case of complex matrices the value EQUATIONSFORM_T assumed as Conjugate transpose.

© 2000–2025, MetaQuotes Ltd.

Orthogonal Factorizations

OpenBLAS предоставляет ряд процедур для разложения общей прямоугольной матрицы A размера m на n в произведение ортогональной (в комплексном случае — унитарной) и треугольной (или, возможно, трапециевидной) матриц.

Реальная матрица Q называется ортогональной, если $Q^T Q = I$; комплексная матрица Q называется унитарной, если $Q^H Q = I$. Ортогональные и унитарные матрицы обладают важным свойством — они сохраняют евклидову норму вектора:

$||x||_2 = ||Qx||_2$, если Q ортогональна или унитарна

Благодаря этому такие матрицы способствуют численной устойчивости, поскольку не усиливают ошибки округления.

Ортогональные разложения широко применяются при решении задач наименьших квадратов. Их также можно использовать для выполнения предварительных шагов при решении задач на собственные значения или сингулярные значения.

Функция	Выполняемое действие
FactorizationQR	Вычисляет QR-разложение общей матрицы размера m на n: A = Q * R. LAPACK-функция GEQRF.
FactorizationQRNonNeg	Вычисляет QR-разложение общей матрицы размера m на n: A = Q * R, где R — верхнетреугольная матрица с неотрицательными элементами на диагонали. LATPACK-функция GEQRFP.
FactorizationQRPivot	Вычисляет QR-разложение общей матрицы размера m на n с перестановкой столбцов: A * P = Q * R. LAPACK-функция GEQP3.
FactorizationLQ	Computes the LQ factorization of a general m-by-n matrix: A = L * Q. LAPACK function GELQF.
FactorizationQL	Computes the QL factorization of a general m-by-n matrix: A = Q * L. LAPACK function GEQLF.
FactorizationRQ	Computes the RQ factorization of a general m-by-n matrix: A = R * Q. LAPACK function GERQF.

© 2000–2025, MetaQuotes Ltd.

FactorizationQR

Computes the QR factorization of a general m-by-n matrix: A = Q * R. LAPACK function GEQRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationQR(
   bool            reduced,      // calculation mode reduced or complete
   matrix&         Q,            // orthogonal matrix Q
   matrix&         R             // upper triangular matrix R
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationQR(
   bool            reduced,      // calculation mode reduced or complete
   matrixf&        Q,            // orthogonal matrix Q
   matrixf&        R             // upper triangular matrix R
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationQR(
   bool            reduced,      // calculation mode reduced or complete
   matrixc&        Q,            // unitary matrix Q
   matrixc&        R             // upper triangular matrix R
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationQR(
   bool            reduced,      // calculation mode reduced or complete
   matrixcf&       Q,            // unitary matrix Q
   matrixcf&       R             // upper triangular matrix R
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices Q, R calculated with dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices Q, R with dimensions (M,M), (M,N).

Q

[out] Orthogonal or unitary matrix Q.

R

[out] Upper triangular matrix R.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

If *reduced* is true

If m >= n, matrix Q is of m-by-n sizes, matrix R is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

If *reduced* is false, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

FactorizationQRNonNeg

Computes the QR factorization of a general m-by-n matrix: A = Q * R. R is an upper triangular matrix with nonnegative diagonal entries. LAPACK function GEQRFP.

Computing for type matrix<double>

```
bool  matrix::FactorizationQRNonNeg(
   bool             reduced,       // calculation mode reduced or complete
   matrix&          Q,             // orthogonal matrix Q
   matrix&          R              // upper triangular matrix R
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationQRNonNeg(
   bool             reduced,       // calculation mode reduced or complete
   matrixf&         Q,             // orthogonal matrix Q
   matrixf&         R              // upper triangular matrix R
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationQRNonNeg(
   bool             reduced,       // calculation mode reduced or complete
   matrixc&         Q,             // unitary matrix Q
   matrixc&         R              // upper triangular matrix R
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationQRNonNeg(
   bool             reduced,       // calculation mode reduced or complete
   matrixcf&        Q,             // unitary matrix Q
   matrixcf&        R              // upper triangular matrix R
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices Q, R calculated with dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices Q, R with dimensions (M,M), (M,N).

Q

[out] Orthogonal or unitary matrix Q.

R

[out] Upper triangular matrix R.

Return Value

© 2000–2025, MetaQuotes Ltd.

Return true if successful, otherwise false in case of an error.

Note

If *reduced* is true

If m >= n, matrix Q is of m-by-n sizes, matrix R is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

If *reduced* is false, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

© 2000–2025, MetaQuotes Ltd.

FactorizationQRPivot

Computes the QR factorization of a general m-by-n matrix with column pivoting: A * P = Q * R. LAPACK function GEQP3.

Computing for type matrix<double>

```
bool  matrix::FactorizationQRPivot(
   bool             reduced,       // calculation mode reduced or complete
   long[]&          jpvt,          // array with predefined permutations
   matrix&          Q,             // orthogonal matrix Q
   matrix&          R,             // upper triangular matrix R
   matrix&          P              // permutation matrix
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationQRPivot(
   bool             reduced,       // calculation mode reduced or complete
   long[]&          jpvt,          // array with predefined permutations
   matrixf&         Q,             // orthogonal matrix Q
   matrixf&         R,             // upper triangular matrix R
   matrixf&         P              // permutation matrix
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationQRPivot(
   bool             reduced,       // calculation mode reduced or complete
   long[]&          jpvt,          // array with predefined permutations
   matrixc&         Q,             // unitary matrix Q
   matrixc&         R,             // upper triangular matrix R
   matrixc&         P              // permutation matrix
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationQRPivot(
   bool             reduced,       // calculation mode reduced or complete
   long[]&          jpvt,          // array with predefined permutations
   matrixcf&        Q,             // unitary matrix Q
   matrixcf&        R,             // upper triangular matrix R
   matrixcf&        P              // permutation matrix
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices Q, R calculated with dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices Q, R with dimensions (M,M), (M,N).

© 2000–2025, MetaQuotes Ltd.

jpvt

[in] Integer array of dimension n. if jpvt(i) ≠ 0, the i-th column of A is moved to the beginning of AP before the computation, and fixed in place during the computation. If jpvt(i) = 0, the i-th column of A is a free column (that is, it may be interchanged during the computation with any other free column). If array has zero size (or not initialized), then all the columns of A assumed to be free.

Q

[out] Orthogonal or unitary matrix Q.

R

[out] Upper triangular matrix R.

P

[out] Permutation matrix P of n-by-n sizes.

Return Value

Return true if successful, otherwise false in case of an error.

Note

If *reduced* is true

If m >= n, matrix Q is of m-by-n sizes, matrix R is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

If *reduced* is false, matrix Q is of m-by-m sizes, matrix R is of m-by-n sizes.

© 2000–2025, MetaQuotes Ltd.

FactorizationLQ

Computes the LQ factorization of a general m-by-n matrix: A = L * Q. LAPACK function GELQF.

Computing for type matrix<double>

```
bool  matrix::FactorizationLQ(
   bool            reduced,      // calculation mode reduced or complete
   matrix&         L,            // lower triangular matrix L
   matrix&         Q             // orthogonal matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationLQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixf&        L,            // lower triangular matrix L
   matrixf&        Q             // orthogonal matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationLQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixc&        L,            // lower triangular matrix L
   matrixc&        Q             // unitary matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationLQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixcf&       L,            // lower triangular matrix L
   matrixcf&       Q             // unitary matrix Q
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices L, Q calculated with reduced dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices L, Q with dimensions (M,N), (N,N).

L

[out] Lower triangular matrix L.

Q

[out] Orthogonal or unitary matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

If *reduced* is true

If m <= n, matrix L is of m-by-m sizes, matrix Q is of m-by-n sizes.

If m > n, matrix L is of m-by-n sizes, matrix Q is of n-by-n sizes.

If *reduced* is false, matrix L is of m-by-n sizes, matrix Q is of n-by-n sizes.

© 2000–2025, MetaQuotes Ltd.

FactorizationQL

Computes the QL factorization of a general m-by-n matrix: A = Q * L. LAPACK function GEQLF.

Computing for type matrix<double>

```
bool  matrix::FactorizationQL(
   bool            reduced,      // calculation mode reduced or complete
   matrix&         Q,            // orthogonal matrix Q
   matrix&         L             // lower triangular matrix L
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationQL(
   bool            reduced,      // calculation mode reduced or complete
   matrixf&        Q,            // orthogonal matrix Q
   matrixf&        L             // lower triangular matrix L
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationQL(
   bool            reduced,      // calculation mode reduced or complete
   matrixc&        Q,            // unitary matrix Q
   matrixc&        L             // lower triangular matrix L
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationQL(
   bool            reduced,      // calculation mode reduced or complete
   matrixcf&       Q,            // unitary matrix Q
   matrixcf&       L             // lower triangular matrix L
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices Q, L calculated with reduced dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices Q, L with dimensions (M,M), (M,N).

Q

[out] Orthogonal or unitary matrix Q.

L

[out] Lower triangular matrix L.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

If *reduced* is true

If m >= n, matrix Q is of m-by-n sizes, matrix L is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix L is of m-by-n sizes.

If *reduced* is false, matrix Q is of m-by-m sizes, matrix L is of m-by-n sizes.

© 2000-2025, MetaQuotes Ltd.

FactorizationRQ

Computes the RQ factorization of a general m-by-n matrix: A = R * Q. LAPACK function GERQF.

Computing for type matrix<double>

```
bool  matrix::FactorizationRQ(
   bool            reduced,      // calculation mode reduced or complete
   matrix&         R,            // upper triangular matrix R
   matrix&         Q             // orthogonal matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationRQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixf&        R,            // upper triangular matrix R
   matrixf&        Q             // orthogonal matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationRQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixc&        R,            // upper triangular matrix R
   matrixc&        Q             // unitary matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationRQ(
   bool            reduced,      // calculation mode reduced or complete
   matrixcf&       R,            // upper triangular matrix R
   matrixcf&       Q             // unitary matrix Q
   );
```

Parameters

reduced

[in] Calculation mode. If reduced is true then matrices R, Q calculated with reduced dimensions (M, K), (K, N). If reduced is false it means complete calculation of matrices L, Q with dimensions (M,N), (N,N).

R

[out] Upper triangular matrix R.

Q

[out] Orthogonal or unitary matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

If *reduced* is true

If m <= n, matrix R is of m-by-m sizes, matrix Q is of m-by-n sizes.

If m > n, matrix R is of m-by-n sizes, matrix Q is of n-by-n sizes.

If *reduced* is false, matrix R is of m-by-n sizes, matrix Q is of n-by-n sizes.

© 2000–2025, MetaQuotes Ltd.

Factorizations

The Factorizations section contains functions for performing various types of matrix factorizations used in numerical solutions of linear systems, stability analysis, and other linear algebra tasks. These factorizations transform the original matrix into simpler forms, making subsequent computations more efficient. All functions are implemented using LAPACK routines and support the types double, float, complex, and complexf.

The functions in this section are used for:

- Preprocessing matrices when solving systems of linear equations;
- Computing determinants, ranks, and matrix inverses;
- Assessing the stability of numerical methods;
- Solving problems in spectral theory and optimization methods.

Matrix factorization is a critical step in many linear algebra algorithms, and this section provides access to the most efficient and well-established factorization techniques.

Function	Action
FactorizationPLU	Computes an LU factorization of a general M-by-N matrix A using partial pivoting with row interchanges. The factorization has the form A = P * L * U. LAPACK function GETRF.
FactorizationPLUQ	Computes an LU factorization of a general N-by-N matrix A with complete pivoting (row and column interchanges). The factorization has the form A = P * L * U * Q where P is a rows permutation matrix, L is lower triangular with unit diagonal elements, U is upper triangular, and Q is a columns permutation matrix. LAPACK function GETC2.
FactorizationPLUGeTrid	Computes an LU factorization of a general (non-symmetric) tridiagonal N-by-N matrix A using elimination with partial pivoting and row interchanges. The factorization has the form A = P * L * U. LAPACK function GTTRF.
FactorizationLDL	Computes the factorization of a real symmetric or complex Hermitian matrix A using the Bunch-Kaufman diagonal pivoting method. LAPACK functions SYTRF, HETRF.
FactorizationLDLComplexSy	Computes the factorization of a complex symmetric (not Hermitian conjugated!) matrix A using the Bunch-Kaufman diagonal pivoting method. LAPACK function SYTRF.
FactorizationLDLSyTridPD	Forms the factorization of a symmetric positive-definite or, for complex data, Hermitian positive-definite tridiagonal matrix A. LAPACK function PTTRF.

© 2000–2025, MetaQuotes Ltd.

Function	Action
FactorizationCholesky	Computes the factorization of a real symmetric or complex Hermitian positive-definite matrix A. LAPACK function POTRF.
FactorizationCholeskySyPS	Computes the Cholesky factorization with complete pivoting of a real symmetric (complex Hermitian) positive semidefinite N-by-N matrix. LAPACK function PSTRF.
FactorizationPLURaw	Computes an LU factorization of a general M-by-N matrix A using partial pivoting with row interchanges. The factorization has the form A = P * L * U where P is a permutation matrix, L is lower triangular with unit diagonal elements (lower trapezoidal if m > n), and U is upper triangular (upper trapezoidal if m < n). LAPACK function GETRF.
FactorizationPLUQRaw	Computes an LU factorization of a general N-by-N matrix A with complete pivoting (row and column interchanges). The factorization has the form A = P * L * U * Q where P is a rows permutation matrix, L is lower triangular with unit diagonal elements, U is upper triangular, and Q is a columns permutation matrix. LAPACK function GETC2.
FactorizationPLUGeTridRaw	Computes an LU factorization of a general (non-symmetric) tridiagonal N-by-N matrix A using elimination with partial pivoting and row interchanges. The factorization has the form A = P * L * U where P is a permutation matrix, L is lower triangular with unit diagonal elements, and U is upper triangular. LAPACK function GTTRF.
FactorizationLDLRaw	Computes the factorization of a real symmetric or complex Hermitian matrix A using the Bunch-Kaufman diagonal pivoting method. LAPACK functions SYTRF, HETRF.

© 2000–2025, MetaQuotes Ltd.

FactorizationPLU

Computes an LU factorization of a general M-by-N matrix A using partial pivoting with row interchanges. The factorization has the form

A = P * L * U

where P is a permutation matrix, L is lower triangular with unit diagonal elements (lower trapezoidal if m > n), and U is upper triangular (upper trapezoidal if m < n). LAPACK function GETRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationPLU(
   matrix&          P,             // permutation matrix P
   matrix&          L,             // lower triangular matrix L
   matrix&          U              // upper triangular matrix U
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationPLU(
   matrixf&         P,             // permutation matrix P
   matrixf&         L,             // lower triangular matrix L
   matrixf&         U              // upper triangular matrix U
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationPLU(
   matrixc&         P,             // permutation matrix P
   matrixc&         L,             // lower triangular matrix L
   matrixc&         U              // upper triangular matrix U
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationPLU(
   matrixcf&        P,             // permutation matrix P
   matrixcf&        L,             // lower triangular matrix L
   matrixcf&        U              // upper triangular matrix U
   );
```

Parameters

P

[out] Permutation matrix P.

L

[out] Lower triangular matrix L with unit diagonal elements.

U

[out] Upper triangular matrix U.

© 2000–2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

FactorizationPLUQ

Computes an LU factorization of a general N-by-N matrix A with complete pivoting (row and column interchanges). The factorization has the form

A = P * L * U * Q

where P is a rows permutation matrix, L is lower triangular with unit diagonal elements, U is upper triangular, and Q is a columns permutation matrix. LAPACK function GETC2.

Computing for type matrix<double>

```
bool  matrix::FactorizationPLUQ(
   matrix&          P,             // rows permutation matrix P
   matrix&          L,             // lower triangular matrix L
   matrix&          U,             // upper triangular matrix U
   matrix&          Q              // cols permutation matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrixf::FactorizationPLUQ(
   matrixf&         P,             // rows permutation matrix P
   matrixf&         L,             // lower triangular matrix L
   matrixf&         U,             // upper triangular matrix U
   matrixf&         Q              // cols permutation matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrixc::FactorizationPLUQ(
   matrixc&         P,             // rows permutation matrix P
   matrixc&         L,             // lower triangular matrix L
   matrixc&         U,             // upper triangular matrix U
   matrixc&         Q              // cols permutation matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::FactorizationPLUQ(
   matrixcf&        P,             // rows permutation matrix P
   matrixcf&        L,             // lower triangular matrix L
   matrixcf&        U,             // upper triangular matrix U
   matrixcf&        Q              // cols permutation matrix Q
   );
```

Parameters

P

[out] Cols permutation matrix P.

L

© 2000–2025, MetaQuotes Ltd.

[out] Lower triangular matrix L with unit diagonal elements.

U

[out] Upper triangular matrix U.

Q

[out] Rows permutation matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

FactorizationPLUGeTrid

Computes an LU factorization of a general (non-symmetric) tridiagonal N-by-N matrix A using elimination with partial pivoting and row interchanges. The factorization has the form

A = P * L * U

where P is a permutation matrix, L is lower triangular with unit diagonal elements, and U is upper triangular. LAPACK function GTTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationPLUGeTrid(
   matrix&          P,             // permutation matrix P
   matrix&          L,             // lower triangular matrix L
   matrix&          U              // upper triangular matrix U
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixf&         P,             // permutation matrix P
   matrixf&         L,             // lower triangular matrix L
   matrixf&         U              // upper triangular matrix U
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixc&         P,             // permutation matrix P
   matrixc&         L,             // lower triangular matrix L
   matrixc&         U              // upper triangular matrix U
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationPLUGeTrid(
   matrixcf&        P,             // permutation matrix P
   matrixcf&        L,             // lower triangular matrix L
   matrixcf&        U              // upper triangular matrix U
   );
```

Parameters

P

[out] Permutation matrix P.

L

[out] Lower triangular matrix L with unit diagonal elements.

U

[out] Upper triangular matrix U.

© 2000–2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

FactorizationLDL

Computes the factorization of a real symmetric or complex Hermitian matrix A using the Bunch-Kaufman diagonal pivoting method. The form of the factorization is:

A = L * D * L**T in case of lower triangular or symmetric matrix A

or

A = U**T * D * U in case of upper triangular matrix A

where L is lower triangular with unit diagonal elements, U is upper triangular with unit diagonal elements. D is a symmetric block-diagonal matrix with 1-by-1 and 2-by-2 diagonal blocks. LAPACK functions SYTRF, HETRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationLDL(
   matrix&          L,             // lower or upper triangular matrix
   matrix&          D              // diagonal matrix D
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationLDL(
   matrixf&         L,             // lower or upper triangular matrix
   matrixf&         D              // diagonal matrix D
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationLDL(
   matrixc&         L,             // lower or upper triangular matrix
   matrixc&         D              // diagonal matrix D
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationLDL(
   matrixcf&        L,             // lower or upper triangular matrix
   matrixcf&        D              // diagonal matrix D
   );
```

Parameters

L

[out] Lower or upper triangular matrix with unit diagonal elements.

D

[out] Symmetric block-diagonal matrix D.

Return Value

© 2000–2025, MetaQuotes Ltd.

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000–2025, MetaQuotes Ltd.

FactorizationLDLComplexSy

Computes the factorization of a complex symmetric (not Hermitian conjugated!) matrix A using the Bunch-Kaufman diagonal pivoting method. The form of the factorization is:

A = L * D * L**T in case of lower triangular or symmetric matrix A

or

A = U**T * D * U in case of upper triangular matrix A

where L is lower triangular with unit diagonal elements, U is upper triangular with unit diagonal elements. D is a symmetric block-diagonal matrix with 1-by-1 and 2-by-2 diagonal blocks. LAPACK function SYTRF.

Computing for type matrix<complex>

```
bool  matrix::FactorizationLDLComplexSy(
   matrixc&        L,             // lower or upper triangular matrix
   matrixc&        D              // diagonal matrix D
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationLDLComplexSy(
   matrixcf&       L,             // lower or upper triangular matrix
   matrixcf&       D              // diagonal matrix D
   );
```

Parameters

L

[out] Lower or upper triangular matrix with unit diagonal elements.

D

[out] Symmetric block-diagonal matrix D.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a complex symmetric, upper triangular or lower triangular matrix. Triangular matrices are assumed to be complex symmetric.

© 2000–2025, MetaQuotes Ltd.

FactorizationLDLSyTridPD

Forms the factorization of a symmetric positive-definite or, for complex data, Hermitian positive-definite tridiagonal matrix A:

A = L * D * L**T in case of lower triangular or symmetric matrix A

or

A = U**T * D * U in case of upper triangular matrix A

where L is lower triangular with unit diagonal elements, U is upper triangular with unit diagonal elements. D is diagonal matrix. LAPACK function PTTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrix&          L,             // lower or upper triangular matrix
   matrix&          D              // diagonal matrix D
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixf&         L,             // lower or upper triangular matrix
   matrixf&         D              // diagonal matrix D
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixc&         L,             // lower or upper triangular matrix
   matrixc&         D              // diagonal matrix D
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationLDLSyTridPD(
   matrixcf&        L,             // lower or upper triangular matrix
   matrixcf&        D              // diagonal matrix D
   );
```

Parameters

L

[out] Lower or upper triangular matrix with unit diagonal elements.

D

[out] Diagonal matrix D.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

Matrices L and D can be used for further calculations with methods LDLSyTridPDLinearEquationsSolution and LDLSyTridPDCondNumReciprocal.

© 2000–2025, MetaQuotes Ltd.

FactorizationCholesky

Computes the factorization of a real symmetric or complex Hermitian positive-definite matrix A. The factorization has the form:

A = L * L**T in case of lower triangular or symmetric matrix A

or

A = U**T * U in case of upper triangular matrix A

where L is lower triangular, U is upper triangular. LAPACK function POTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationCholesky(
   matrix&          L              // lower or upper triangular matrix
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationCholesky(
   matrixf&         L              // lower or upper triangular matrix
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationCholesky(
   matrixc&         L              // lower or upper triangular matrix
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationCholesky(
   matrixcf&        L              // lower or upper triangular matrix
   );
```

Parameters

L

[out] Lower or upper triangular matrix.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

Matrices L and D can be used for further calculations with methods LDLSyTridPDLinearEquationsSolution and LDLSyTridPDCondNumReciprocal.

© 2000–2025, MetaQuotes Ltd.

FactorizationCholeskySyPS

Computes the Cholesky factorization with complete pivoting of a real symmetric (complex Hermitian) positive semidefinite N-by-N matrix. The form of the factorization is:

P**T * A * P = L * L**T in case of lower triangular or symmetric matrix A

or

P**T * A * P = U**T * U in case of upper triangular matrix A

where P is a permutation matrix, L is lower triangular, U is upper triangular. LAPACK function PSTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationCholeskySyPS(
   double          tol           // tolerance
   matrix&         P,            // permutation matrix P
   matrix&         L             // lower or upper triangular matrix
   );
```

Computing for type matrix<float>

```
bool  matrix::FactorizationCholeskySyPS(
   float           tol           // tolerance
   matrixf&        P,            // permutation matrix P
   matrixf&        L             // lower or upper triangular matrix
   );
```

Computing for type matrix<complex>

```
bool  matrix::FactorizationCholeskySyPS(
   double          tol           // tolerance
   matrixc&        P,            // permutation matrix P
   matrixc&        L             // lower or upper triangular matrix
   );
```

Computing for type matrix<complexf>

```
bool  matrix::FactorizationCholeskySyPS(
   float           tol           // tolerance
   matrixcf&       P,            // permutation matrix P
   matrixcf&       L             // lower or upper triangular matrix
   );
```

Parameters

tol

[in] User defined tolerance. If tol < 0, then n*ε*max(A[k,k]), where ε is the machine precision, will be used. The algorithm terminates at the (k-1)st step, if the pivot <=tol.

P

[out] Permutation matrix P.

© 2000–2025, MetaQuotes Ltd.

L

[out] Lower or upper triangular matrix.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

© 2000–2025, MetaQuotes Ltd.

FactorizationPLURaw

Computes an LU factorization of a general M-by-N matrix A using partial pivoting with row interchanges. The factorization has the form

A = P * L * U

where P is a permutation matrix, L is lower triangular with unit diagonal elements (lower trapezoidal if m > n), and U is upper triangular (upper trapezoidal if m < n). LAPACK function GETRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationPLURaw(
   matrix&          AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<float>

```
bool  matrixf::FactorizationPLURaw(
   matrixf&         AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<complex>

```
bool  matrixc::FactorizationPLURaw(
   matrixc&         AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::FactorizationPLURaw(
   matrixcf&        AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Parameters

AF

[out] Factored matrix A. The factors L and U from the factorization A = P*L*U; the unit diagonal elements of L are not stored.

ipiv

[out] Pivot indices array of size M; row i of the matrix A was interchanged with row ipiv[i].

Return Value

Return true if successful, otherwise false in case of an error.

Note

© 2000–2025, MetaQuotes Ltd.

Matrix AF and pivot indices array ipiv[] are raw output of the GETRF function and can be used for further calculations with methods PLULinearEquationsSolution, PLUInverse and PLUCondNumReciprocal. In these cases original matrix A must be square.

© 2000–2025, MetaQuotes Ltd.

FactorizationPLUQRaw

Computes an LU factorization of a general N-by-N matrix A with complete pivoting (row and column interchanges). The factorization has the form

A = P * L * U * Q

where P is a rows permutation matrix, L is lower triangular with unit diagonal elements, U is upper triangular, and Q is a columns permutation matrix. LAPACK function GETC2.

Computing for type matrix<double>

```
bool  matrix::FactorizationPLUQRaw(
   matrix&          AF,            // factored matrix A
   long[]&          ipiv,          // pivot indices array
   long[]&          jpiv           // pivot indices array
   );
```

Computing for type matrix<float>

```
bool  matrixf::FactorizationPLUQRaw(
   matrixf&         AF,            // factored matrix A
   long[]&          ipiv,          // pivot indices array
   long[]&          jpiv           // pivot indices array
   );
```

Computing for type matrix<complex>

```
bool  matrixc::FactorizationPLUQRaw(
   matrixc&         AF,            // factored matrix A
   long[]&          ipiv,          // pivot indices array
   long[]&          jpiv           // pivot indices array
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::FactorizationPLUQRaw(
   matrixcf&        AF,            // factored matrix A
   long[]&          ipiv,          // pivot indices array
   long[]&          jpiv           // pivot indices array
   );
```

Parameters

AF

[out] Factored matrix A. The factors L and U from the factorization A = P*L*U*Q; the unit diagonal elements of L are not stored.

ipiv

[out] Pivot indices array of size N; row i of the matrix A was interchanged with row ipiv[i].

jpiv

© 2000–2025, MetaQuotes Ltd.

[out] Pivot indices array of size N; column j of the matrix A was interchanged with column jpiv[j].

Return Value

Return true if successful, otherwise false in case of an error.

Note

Matrix AF and pivot indices arrays ipiv[] and jpiv[] are raw output of the GETC2 function and can be used for further calculations with method PLUQLinearEquationsSolution.

© 2000–2025, MetaQuotes Ltd.

FactorizationPLUGeTridRaw

Computes an LU factorization of a general (non-symmetric) tridiagonal N-by-N matrix A using elimination with partial pivoting and row interchanges. The factorization has the form

A = P * L * U

where P is a permutation matrix, L is lower triangular with unit diagonal elements, and U is upper triangular. LAPACK function GTTRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationPLUGeTridRaw(
   matrix&          AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<float>

```
bool  matrixf::FactorizationPLUGeTridRaw(
   matrixf&         AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<complex>

```
bool  matrixc::FactorizationPLUGeTridRaw(
   matrixc&         AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::FactorizationPLUGeTridRaw(
   matrixcf&        AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Parameters

AF

[out] Factored matrix A. The factors L and U from the factorization A = P*L*U; the unit diagonal elements of L are not stored.

ipiv

[out] Pivot indices array of size N; row i of the matrix A was interchanged with row ipiv[i].

Return Value

Return true if successful, otherwise false in case of an error.

Note

© 2000–2025, MetaQuotes Ltd.

Matrix AF and pivot indices array ipiv[] are raw output of the GTTRF function and can be used for further calculations with methods PLUGeTridLinearEquationsSolution and PLUGeTridCondNumReciprocal.

© 2000–2025, MetaQuotes Ltd.

FactorizationLDLRaw

Computes the factorization of a real symmetric or complex Hermitian matrix A using the Bunch-Kaufman diagonal pivoting method. The form of the factorization is:

A = L * D * L**T in case of lower triangular or symmetric matrix A

or

A = U**T * D * U in case of upper triangular matrix A

where L is lower triangular with unit diagonal elements, U is upper triangular with unit diagonal elements. D is a symmetric block-diagonal matrix with 1-by-1 and 2-by-2 diagonal blocks. LAPACK functions SYTRF, HETRF.

Computing for type matrix<double>

```
bool  matrix::FactorizationLDLRaw(
   matrix&          AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<float>

```
bool  matrixf::FactorizationLDLRaw(
   matrixf&         AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<complex>

```
bool  matrixc::FactorizationLDLRaw(
   matrixc&         AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::FactorizationLDLRaw(
   matrixcf&        AF,            // factored matrix A
   long[]&          ipiv           // pivot indices array
   );
```

Parameters

AF

[out] Factored matrix A. The block diagonal matrix D and factor L or U.

ipiv

[out] Pivot indices array of size N; details of the interchanges and the block structure of D.

Return Value

© 2000–2025, MetaQuotes Ltd.

Return true if successful, otherwise false in case of an error.

Note

The input can be a symmetric (Hermitian), upper triangular or lower triangular matrix. Triangular matrices are assumed to be symmetric (Hermitian conjugated).

Matrix AF and pivot indices array ipiv[] are raw output of the SYTRF (HETRF) function and can be used for further calculations with methods LDLLinearEquationsSolution, LDLInverse and LDLCondNumReciprocal.

© 2000–2025, MetaQuotes Ltd.

Factored Calculations

Этот раздел справки описывает набор функций для численного решения задач линейной алгебры, основанных на факторизациях матриц и методах библиотеки LAPACK, обёрнутых в классы matrix, matrixf, matrixc и matrixcf.

Функции ориентированы на высокоэффективные и надёжные вычисления, применимые к различным типам матриц: вещественным и комплексным, одинарной и двойной точности. Они охватывают ключевые задачи прикладной линейной алгебры, включая:

- Решение систем линейных уравнений (СЛАУ);
- Оценку числа обусловленности матрицы;
- Обратные преобразования и вычисление обратных матриц;
- Специальные приёмы на основе SVD для получения псевдообратных и полярных разложений.

Функции работают в связке с предварительными матричными факторизациями, такими как Cholesky, LDL, PLU и другими, реализованными в соответствующих модулях. Используемые алгоритмы повторяют поведение LAPACK и позволяют эффективно обрабатывать как плотные, так и специализированные структуры матриц (например, тридиагональные).

Ниже представлены все доступные функции, сгруппированные по используемому методу факторизации.

Функция	Выполняемое действие
PLULinearEquationsSolution	Solves a system of linear equations A * X = B, A**T * X = B, or A**H * X = B with an LU-factored square coefficient matrix AF computed by FactorizationPLURaw, with multiple right-hand sides. LAPACK function GETRS.
PLUInverse	Computes the inverse of an LU-factored general matrix AF computed by FactorizationPLURaw. LAPACK function GETRI.
PLUCondNumReciprocal	Estimates the reciprocal of the condition number of a general matrix A in either the one-norm or infinity-norm, using the LU factorization computed by FactorizationPLURaw. LAPACK function GECON.
PLUQLinearEquationsSolution	Solves a system of linear equations A * X = scale * RHS with a general N-by-N matrix A using the LU-factoization with complete pivoting computed by FactorizationPLUQRaw. LAPACK function GESC2.
PLUGeTridLinearEquationsSolution	Solves a system of linear equations A * X = B, A**T * X = B, or A**H * X = B with a tridiagonal matrix A using the LU-factorization computed by FactorizationPLUGeTridRaw, with multiple right-hand sides. LAPACK function GTTRS.
PLUGeTridCondNumReciprocal	Estimates the reciprocal of the condition number of a general tridiagonal matrix A in either the one-norm

© 2000–2025, MetaQuotes Ltd.

Функция	Выполняемое действие
	or infinity-norm, using the LU factorization computed by FactorizationPLUGeTridRaw. LAPACK function GTCON.
LDLLinearEquationsSolution	Solves a system of linear equations A * X = B with a real symmetric or complex Hermitian indefinite matrix using the factorization A = U**T * D * U or A = L * D * L**T computed by FactorizationLDLRaw, with multiple right-hand sides. LAPACK functions SYTRS, HETRS.
LDLInverse	Computes the inverse of a real symmetric or complex Hermitian indefinite matrix using the factorization A = U**T * D * U or A = L * D * L**T computed by FactorizationLDLRaw. LAPACK functions SYTRI, HETRI.
LDLCondNumReciprocal	Estimates the reciprocal of the condition number of a real symmetric or complex Hermitian indefinite matrix A, using the LDLT factorization computed by FactorizationLDLRaw. LAPACK functions SYCON, HECON.
LDLSyTridPDLinearEquationsSolution	Solves a system of linear equations A * X = B with a real symmetric or complex Hermitian positive-definite tridiagonal matrix using the factorization A = L * D * L**T computed by FactorizationLDLSyTridPD, with multiple right-hand sides. LAPACK function PTTRS.
LDLSyTridPDCondNumReciprocal	Estimates the reciprocal of the condition number of a real symmetric or complex Hermitian positive-definite tridiagonal matrix A using the LDLT factorization computed by FactorizationLDLSyTridPD. LAPACK function PTCON.
CholeskyLinearEquationsSolution	Solves a system of linear equations A * X = B with a real symmetric or complex Hermitian positive-definite matrix using the factorization A = L * L**T computed by FactorizationCholesky, with multiple right-hand sides. LAPACK function POTRS.
CholeskyInverse	Computes the inverse of a real symmetric or complex Hermitian positive-definite matrix using the LLT factorization computed by FactorizationCholesky. LAPACK function POTRI.
CholeskyCondNumReciprocal	Estimates the reciprocal of the condition number of a real symmetric or complex Hermitian positive-definite matrix A using the LDL factorization computed by FactorizationCholesky. LAPACK function POCON.

© 2000–2025, MetaQuotes Ltd.

Функция	Выполняемое действие
SylvesterEquationSchur	Solves Sylvester equation for real quasi-triangular or complex triangular matrices: A*X + X*B = C where A and B are both upper triangular. A is m-by-m and B is n-by-n; the right hand side C and the solution X are m-by-n. LAPACK function TRSYL.
SylvesterEquationSchurBlocked	Solves Sylvester equation for real quasi-triangular or complex triangular matrices: A*X + X*B = C where A and B are both upper triangular. A is m-by-m and B is n-by-n; the right hand side C and the solution X are m-by-n. LAPACK function TRSYL3. This is the block (BLAS level 3) version of TRSYL. Faster up to 5 times but not so accurate.
Pseudo Inverse	There is no special function in the OpenBLAS library to calculate the pseudo-inverse of a matrix. However, for this purpose can be used singular value decomposition (SVD)
Polar Decomposition	There is no special function in the OpenBLAS library to calculate the polar decomposition of a matrix. However, for this purpose can be used singular value decomposition (SVD)

Функции обеспечивают поддержку различных типов матриц (вещественных, комплексных, одинарной и двойной точности) и охватывают следующие ключевые задачи.

Оценка числа обусловленности (Condition Number Estimation)

В отличие от истинного числа обусловленности, оценка производится приблизительно, но гораздо быстрее. Здесь *CondNumReciprocal* обозначает обратное инвертированное число обусловенности (то есть 1/cond_number).

Функции типа **CondNumReciprocal* предназначены для оценки обратной величины числа обусловленности матрицы:

- CholeskyCondNumReciprocal — для положительно определённых матриц с факторизацией LLT;
- LDLCondNumReciprocal — для неопределённых симметричных или эрмитовых матриц с факторизацией LDL;
- LDLSyTridPDCondNumReciprocal — для положительно определённых симметричных тридиагональных матриц;
- PLUCondNumReciprocal, PLUGeTridCondNumReciprocal — для обобщённых и тридиагональных матриц с LU-факторизацией.

Обратные матрицы (Matrix Inverse)

Функции типа **Inverse* вычисляют обратную матрицу на основе соответствующей факторизации:

- CholeskyInverse, LDLInverse, PLUInverse — для различных форм факторизаций;
- используют массив индексов ipiv, если требуется информация о перестановках.

© 2000–2025, MetaQuotes Ltd.

Решение СЛАУ (Linear System Solvers)

Функции *LinearEquationsSolution решают СЛАУ вида A * X = B:

- Поддерживаются как матричные, так и векторные правые части;
- Учитывается возможность транспонирования (или эрмитового сопряжения) матрицы коэффициентов через ENUM_EQUATIONS_FORM;
- Отдельные функции предусмотрены для тридиагональных и обобщённых матриц.

Поддержка расширенных LU-факторизаций

Функция PLUQLinearEquationsSolution решает систему с масштабированием:

```
A * X = scale * B
```

на основе полной LU-факторизации с перестановками по строкам и столбцам.

Специальные методы (на основе SVD)

Хотя функции библиотеки напрямую не реализуют полярное разложение или псевдообратную матрицу, раздел содержит примеры реализации на базе сингулярного разложения:

- PolarDecomposition — разложение A = Q * P, где Q — ортогональная, P — симметричная положительно определённая матрица;
- PseudoInverse — вычисление псевдообратной матрицы A+ = V * Σ⁺ * U□.

Примечания

- Все функции возвращают true при успехе, false при ошибке.
- Необходимыми предпосылками являются факторизации, выполненные заранее (FactorizationCholesky, FactorizationLDLRaw, GETRF и т.д.).
- Результаты вычислений могут быть чувствительны к численной стабильности, особенно при малых значениях *rcond*.

© 2000–2025, MetaQuotes Ltd.

PLULinearEquationsSolution

Solves a system of linear equations

A * X = B, A**T * X = B, or A**H * X = B

with an LU-factored square coefficient matrix AF computed by FactorizationPLURaw, with multiple right-hand sides. LAPACK function GETRS.

Computing for type matrix<double>

```
bool  matrix::PLULinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   matrix&              B,             // right hand side matrix B
   matrix&              X              // solution matrix X
   );

bool  matrix::PLULinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   vector&              B,             // right hand side vector B
   vector&              X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrixf::PLULinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   matrixf&             B,             // right hand side matrix B
   matrixf&             X              // solution matrix X
   );

bool  matrixf::PLULinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   vectorf&             B,             // right hand side vector B
   vectorf&             X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrixc::PLULinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   matrixc&             B,             // right hand side matrix B
   matrixc&             X              // solution matrix X
   );
```

© 2000–2025, MetaQuotes Ltd.

```
bool  matrixc::LinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   vectorc&             B,             // right hand side vector B
   vectorc&             X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::PLULinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   matrixcf&            B,             // right hand side matrix B
   matrixcf&            X              // solution matrix X
   );

bool  matrixcf::PLULinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   vectorcf&            B,             // right hand side vector B
   vectorcf&            X              // solution vector X
   );
```

Parameters

ipiv

[in] Pivot indices array obtained as result of GETRF function.

trans

[in] ENUM_EQUATIONS_FORM enumeration value which specifies the form of the system of equations.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of GETRF function.

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000-2025, MetaQuotes Ltd.

ENUM_EQUATIONS_FORM

An enumeration defining which form of the equations' system calculated.

ID	Description
EQUATIONSFORM_N	'N': A * X = B (No transpose)
EQUATIONSFORM_T	'T': A**T * X = B (Transpose)
EQUATIONSFORM_C	'C': A**H * X = B (Conjugate transpose)

In case of real matrices the value EQUATIONSFORM_C assumed as Transpose.

© 2000–2025, MetaQuotes Ltd.

PLUInverse

Computes the inverse of an LU-factored general matrix AF computed by FactorizationPLURaw. LAPACK function GETRI.

Computing for type matrix<double>

```
bool  matrix::PLUInverse(
   long[]&          ipiv,           // pivot indices array
   matrix&          AI              // inverse of matrix A
   );
```

Computing for type matrix<float>

```
bool  matrixf::PLUInverse(
   long[]&          ipiv,           // pivot indices array
   matrixf&         AI              // inverse of matrix A
   );
```

Computing for type matrix<complex>

```
bool  matrixc::PLUInverse(
   long[]&          ipiv,           // pivot indices array
   matrixc&         AI              // inverse of matrix A
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::PLUInverse(
   long[]&          ipiv,           // pivot indices array
   matrixcf&        AI              // inverse of matrix A
   );
```

Parameters

ipiv

[in] Pivot indices array obtained as result of GETRF function.

AI

[out] Inverted matrix.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of GETRF function.

© 2000–2025, MetaQuotes Ltd.

PLUCondNumReciprocal

Estimates the reciprocal of the condition number of a general matrix A in either the one-norm or infinity-norm, using the LU factorization computed by FactorizationPLURaw. LAPACK function GECON.

Computing for type matrix<double>

```
bool  matrix::PLUCondNumReciprocal(
   ENUM_BLAS_NORM  norm,          // matrix norm
   double          anorm,         // matrix norm value
   double&         rcond          // condition number reciprocal
   );
```

Computing for type matrix<float>

```
bool  matrixf::PLUCondNumReciprocal(
   ENUM_BLAS_NORM  norm,          // matrix norm
   float           anorm,         // matrix norm value
   float&          rcond          // condition number reciprocal
   );
```

Computing for type matrix<complex>

```
bool  matrixc::PLUCondNumReciprocal(
   ENUM_BLAS_NORM  norm,          // matrix norm
   double          anorm,         // matrix norm value
   double&         rcond          // condition number reciprocal
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::PLUCondNumReciprocal(
   ENUM_BLAS_NORM  norm,          // matrix norm
   float           anorm,         // matrix norm value
   float&          rcond          // condition number reciprocal
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORM enumeration, which defines estimation the reciprocal of the condition number.

anorm

[in] Matrix norm value. If *norm* = 'O', the one-norm of the original matrix A. If *norm* = 'I', the infinity-norm of the original matrix A. Norm value can be obtained with MatrixNorm of the original matrix A.

rcond

[out] An estimate of the reciprocal of the condition number. The routine sets rcond=0 if the estimate underflows; in this case the matrix is singular (to working precision). However, anytime rcond is small compared to 1.0, for the working precision, the matrix may be poorly conditioned or even singular.

© 2000–2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of GETRF function.

The computed *rcond* is never less than r (the reciprocal of the true condition number) and in practice is nearly always less than 10r. A call to this routine involves solving a number of systems of linear equations A*x = b; the number is usually 4 or 5 and never more than 11

ENUM_BLAS_NORM

An enumeration defining the matrix norm.

ID	Description
BLASNORM_O	'O': One-norm
BLASNORM_I	'I': Infinity-norm

© 2000–2025, MetaQuotes Ltd.

PLUQLinearEquationsSolution

Solves a system of linear equations

A * X = scale * RHS

with a general N-by-N matrix A using the LU-factoization with complete pivoting computed by FactorizationPLUQRaw. LAPACK function GESC2.

Computing for type matrix<double>

```
bool  matrix::PLUQLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   long[]&              jpiv,          // pivot indices array
   matrix&              B,             // right hand side matrix B
   matrix&              X,             // solution matrix X
   double&              scale          // scale factor
   );

bool  matrix::PLUQLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   long[]&              jpiv,          // pivot indices array
   vector&              B,             // right hand side vector B
   vector&              X,             // solution vector X
   double&              scale          // scale factor
   );
```

Computing for type matrix<float>

```
bool  matrixf::PLUQLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   long[]&              jpiv,          // pivot indices array
   matrixf&             B,             // right hand side matrix B
   matrixf&             X,             // solution matrix X
   float&               scale          // scale factor
   );

bool  matrixf::PLUQLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   long[]&              jpiv,          // pivot indices array
   vectorf&             B,             // right hand side vector B
   vectorf&             X,             // solution vector X
   float&               scale          // scale factor
   );
```

Computing for type matrix<complex>

```
bool  matrixc::PLUQLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   long[]&              jpiv,          // pivot indices array
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixc&             B,             // right hand side matrix B
   matrixc&             X,             // solution matrix X
   double&              scale          // scale factor
   );

bool  matrixc::PLUQLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   long[]&              jpiv,          // pivot indices array
   vectorc&             B,             // right hand side vector B
   vectorc&             X,             // solution vector X
   double&              scale          // scale factor
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::PLUQLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   long[]&              jpiv,          // pivot indices array
   matrixcf&            B,             // right hand side matrix B
   matrixcf&            X,             // solution matrix X
   float&               scale          // scale factor
   );

bool  matrixcf::PLUQLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   long[]&              jpiv,          // pivot indices array
   vectorcf&            B,             // right hand side vector B
   vectorcf&            X,             // solution vector X
   float&               scale          // scale factor
   );
```

Parameters

ipiv

[in] Rows pivot indices array obtained as result of GETC2 function.

jpiv

[in] Columns pivot indices array obtained as result of GETC2 function.

B

[in] Matrix B whose column is the right-hand side for the system of equations, matix B must contain only one column. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

scale

[out] Scale factor; *scale* is chosen 0 <= *scale* <= 1 to prevent overflow in the solution.

© 2000–2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of GETC2 function.

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

PLUGeTridLinearEquationsSolution

Solves a system of linear equations

A * X = B, A**T * X = B, or A**H * X = B

with a tridiagonal matrix A using the LU-factorization computed by FactorizationPLUGeTridRaw, with multiple right-hand sides. LAPACK function GTTRS.

Computing for type matrix<double>

```
bool  matrix::PLUGeTridLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   matrix&              B,             // right hand side matrix B
   matrix&              X              // solution matrix X
   );

bool  matrix::PLUGeTridLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   vector&              B,             // right hand side vector B
   vector&              X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrixf::PLUGeTridLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   matrixf&             B,             // right hand side matrix B
   matrixf&             X              // solution matrix X
   );

bool  matrixf::PLUGeTridLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   vectorf&             B,             // right hand side vector B
   vectorf&             X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrixc::PLUGeTridLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   matrixc&             B,             // right hand side matrix B
   matrixc&             X              // solution matrix X
   );
```

© 2000–2025, MetaQuotes Ltd.

```
bool  matrixc::PLUGeTridLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   vectorc&             B,             // right hand side vector B
   vectorc&             X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::PLUGeTridLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   matrixcf&            B,             // right hand side matrix B
   matrixcf&            X              // solution matrix X
   );

bool  matrixcf::PLUGeTridLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   ENUM_EQUATIONS_FORM  trans,         // form of the system of equations
   vectorcf&            B,             // right hand side vector B
   vectorcf&            X              // solution vector X
   );
```

Parameters

ipiv

[in] Pivot indices array obtained as result of GTTRF function.

trans

[in] ENUM_EQUATIONS_FORM enumeration value which specifies the form of the system of equations.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of FactorizationPLUGeTridRaw method.

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

ENUM_EQUATIONS_FORM

An enumeration defining which form of the equations' system calculated.

ID	Description
EQUATIONSFORM_N	'N': A * X = B (No transpose)
EQUATIONSFORM_T	'T': A**T * X = B (Transpose)
EQUATIONSFORM_C	'C': A**H * X = B (Conjugate transpose)

In case of real matrices the value EQUATIONSFORM_C assumed as Transpose.

© 2000–2025, MetaQuotes Ltd.

PLUGeTridCondNumReciprocal

Estimates the reciprocal of the condition number of a general tridiagonal matrix A in either the one-norm or infinity-norm, using the LU factorization computed by FactorizationPLUGeTridRaw. LAPACK function GTCON.

Computing for type matrix<double>

```
bool  matrix::PLUGeTridCondNumReciprocal(
   long[]&          ipiv,          // pivot indices array
   ENUM_BLAS_NORM   norm,          // matrix norm
   double           anorm,         // matrix norm value
   double&          rcond          // condition number reciprocal
   );
```

Computing for type matrix<float>

```
bool  matrixf::PLUGeTridCondNumReciprocal(
   long[]&          ipiv,          // pivot indices array
   ENUM_BLAS_NORM   norm,          // matrix norm
   float            anorm,         // matrix norm value
   float&           rcond          // condition number reciprocal
   );
```

Computing for type matrix<complex>

```
bool  matrixc::PLUGeTridCondNumReciprocal(
   long[]&          ipiv,          // pivot indices array
   ENUM_BLAS_NORM   norm,          // matrix norm
   double           anorm,         // matrix norm value
   double&          rcond          // condition number reciprocal
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::PLUGeTridCondNumReciprocal(
   long[]&          ipiv,          // pivot indices array
   ENUM_BLAS_NORM   norm,          // matrix norm
   float            anorm,         // matrix norm value
   float&           rcond          // condition number reciprocal
   );
```

Parameters

ipiv

[in] Pivot indices array obtained as result of GTTRF function.

norm

[in] Value from the ENUM_BLAS_NORM enumeration, which defines estimation the reciprocal of the condition number.

anorm

© 2000–2025, MetaQuotes Ltd.

[in] Matrix norm value. If *norm* = 'O', the one-norm of the original matrix A. If *norm* = 'I', the infinity-norm of the original matrix A. Norm value can be obtained with MatrixNormGeTrid of the original matrix A.

rcond

[out] An estimate of the reciprocal of the condition number. The routine sets rcond=0 if the estimate underflows; in this case the matrix is singular (to working precision). However, anytime rcond is small compared to 1.0, for the working precision, the matrix may be poorly conditioned or even singular.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of FactorizationPLUGeTridRaw method.

The computed *rcond* is never less than r (the reciprocal of the true condition number) and in practice is nearly always less than 10r. A call to this routine involves solving a number of systems of linear equations A*x = b; the number is usually 4 or 5 and never more than 11

ENUM_BLAS_NORM

An enumeration defining the matrix norm.

ID	Description
BLASNORM_O	'O': One-norm
BLASNORM_I	'I': Infinity-norm

© 2000–2025, MetaQuotes Ltd.

LDLLinearEquationsSolution

Solves a system of linear equations A * X = B with a real symmetric or complex Hermitian indefinite matrix using the factorization A = U**T * D * U or A = L * D * L**T computed by FactorizationLDLRaw, with multiple right-hand sides. LAPACK functions SYTRS, HETRS.

Computing for type matrix<double>

```
bool  matrix::LDLLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   matrix&              B,             // right hand side matrix B
   matrix&              X              // solution matrix X
   );

bool  matrix::LDLLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   vector&              B,             // right hand side vector B
   vector&              X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrixf::LDLLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   matrixf&             B,             // right hand side matrix B
   matrixf&             X              // solution matrix X
   );

bool  matrixf::LDLLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   vectorf&             B,             // right hand side vector B
   vectorf&             X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LDLLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   matrixc&             B,             // right hand side matrix B
   matrixc&             X              // solution matrix X
   );

bool  matrixc::LDLLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   vectorc&             B,             // right hand side vector B
   vectorc&             X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LDLLinearEquationsSolution(
```

© 2000–2025, MetaQuotes Ltd.

```
   long[]&              ipiv,          // pivot indices array
   matrixcf&            B,             // right hand side matrix B
   matrixcf&            X              // solution matrix X
   );

bool  matrixcf::LDLLinearEquationsSolution(
   long[]&              ipiv,          // pivot indices array
   vectorcf&            B,             // right hand side vector B
   vectorcf&            X              // solution vector X
   );
```

Parameters

ipiv

[in] Pivot indices array obtained as result of SYTRF or HETRF function.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of SYTRF or HETRF function.

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

LDLInverse

Computes the inverse of a real symmetric or complex Hermitian indefinite matrix using the factorization A = U**T * D * U or A = L * D * L**T computed by FactorizationLDLRaw. LAPACK functions SYTRI, HETRI.

Computing for type matrix<double>

```
bool  matrix::LDLInverse(
   long[]&          ipiv,           // pivot indices array
   matrix&          AI              // inverse of matrix A
   );
```

Computing for type matrix<float>

```
bool  matrixf::LDLInverse(
   long[]&          ipiv,           // pivot indices array
   matrixf&         AI              // inverse of matrix A
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LDLInverse(
   long[]&          ipiv,           // pivot indices array
   matrixc&         AI              // inverse of matrix A
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LDLInverse(
   long[]&          ipiv,           // pivot indices array
   matrixcf&        AI              // inverse of matrix A
   );
```

Parameters

ipiv

[in] Pivot indices array obtained as result of SYTRF or HETRF function.

AI

[out] Inverted matrix.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of SYTRF or HETRF function.

© 2000–2025, MetaQuotes Ltd.

LDLCondNumReciprocal

Estimates the reciprocal of the condition number of a real symmetric or complex Hermitian indefinite matrix A, using the LDLT factorization computed by FactorizationLDLRaw. LAPACK functions SYCON, HECON.

Computing for type matrix<double>

```
bool  matrix::LDLCondNumReciprocal(
   long[]&          ipiv,          // pivot indices array
   double           anorm,         // matrix norm value
   double&          rcond          // condition number reciprocal
   );
```

Computing for type matrix<float>

```
bool  matrixf::LDLCondNumReciprocal(
   long[]&          ipiv,          // pivot indices array
   float            anorm,         // matrix norm value
   float&           rcond          // condition number reciprocal
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LDLCondNumReciprocal(
   long[]&          ipiv,          // pivot indices array
   double           anorm,         // matrix norm value
   double&          rcond          // condition number reciprocal
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LDLCondNumReciprocal(
   long[]&          ipiv,          // pivot indices array
   float            anorm,         // matrix norm value
   float&           rcond          // condition number reciprocal
   );
```

Parameters

ipiv

[in] Pivot indices array obtained as result of SYTRF or HETRF function.

anorm

[in] Matrix one-norm value. Norm value can be obtained with MatrixNormSy of the original matrix A.

rcond

[out] An estimate of the reciprocal of the condition number. The routine sets rcond=0 if the estimate underflows; in this case the matrix is singular (to working precision). However, anytime rcond is small compared to 1.0, for the working precision, the matrix may be poorly conditioned or even singular.

© 2000–2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix AF obtained as result of SYTRF or HETRF function.

The computed *rcond* is never less than r (the reciprocal of the true condition number) and in practice is nearly always less than 10r. A call to this routine involves solving a number of systems of linear equations A*x = b; the number is usually 4 or 5 and never more than 11

© 2000–2025, MetaQuotes Ltd.

LDLSyTridPDLinearEquationsSolution

Solves a system of linear equations A * X = B with a real symmetric or complex Hermitian positive-definite tridiagonal matrix using the factorization A = L * D * L**T computed by FactorizationLDLSyTridPD, with multiple right-hand sides. LAPACK function PTTRS.

Computing for type matrix<double>

```
bool  matrix::LDLSyTridPDLinearEquationsSolution(
   matrix&              D,             // diagonal matrix
   matrix&              B,             // right hand side matrix B
   matrix&              X              // solution matrix X
   );

bool  matrix::LDLSyTridPDLinearEquationsSolution(
   matrix&              D,             // diagonal matrix
   vector&              B,             // right hand side vector B
   vector&              X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrixf::LDLSyTridPDLinearEquationsSolution(
   matrixf&             D,             // diagonal matrix
   matrixf&             B,             // right hand side matrix B
   matrixf&             X              // solution matrix X
   );

bool  matrixf::LDLSyTridPDLinearEquationsSolution(
   matrixf&             D,             // diagonal matrix
   vectorf&             B,             // right hand side vector B
   vectorf&             X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LDLSyTridPDLinearEquationsSolution(
   matrixcd&            D,             // diagonal matrix
   matrixc&             B,             // right hand side matrix B
   matrixc&             X              // solution matrix X
   );

bool  matrixc::LDLSyTridPDLinearEquationsSolution(
   matrixcd&            D,             // diagonal matrix
   vectorc&             B,             // right hand side vector B
   vectorc&             X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LDLSyTridPDLinearEquationsSolution(
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixcf&            D,             // diagonal matrix
   matrixcf&            B,             // right hand side matrix B
   matrixcf&            X              // solution matrix X
   );

bool  matrixcf::LDLSyTridPDLinearEquationsSolution(
   matrixcf&            D,             // diagonal matrix
   vectorcf&            B,             // right hand side vector B
   vectorcf&            X              // solution vector X
   );
```

Parameters

D

[in] Diagonal matrix obtained as result of FactorizationLDLSyTridPD method.

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix L obtained as result of FactorizationLDLSyTridPD method.

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

LDLSyTridPDCondNumReciprocal

Estimates the reciprocal of the condition number of a real symmetric or complex Hermitian positive-definite tridiagonal matrix A using the LDLT factorization computed by FactorizationLDLSyTridPD. LAPACK function PTCON.

Computing for type matrix<double>

```
bool  matrix::LDLSyTridPDCondNumReciprocal(
   matrix&          D,              // diagonal matrix
   double           anorm,          // matrix norm value
   double&          rcond           // condition number reciprocal
   );
```

Computing for type matrix<float>

```
bool  matrixf::LDLSyTridPDCondNumReciprocal(
   matrixf&         D,              // diagonal matrix
   float            anorm,          // matrix norm value
   float&           rcond           // condition number reciprocal
   );
```

Computing for type matrix<complex>

```
bool  matrixc::LDLSyTridPDCondNumReciprocal(
   matrixc&         D,              // diagonal matrix
   double           anorm,          // matrix norm value
   double&          rcond           // condition number reciprocal
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::LDLSyTridPDCondNumReciprocal(
   matrixcf&        D,              // diagonal matrix
   float            anorm,          // matrix norm value
   float&           rcond           // condition number reciprocal
   );
```

Parameters

D

[in] Diagonal matrix obtained as result of FactorizationLDLSyTridPD method.

anorm

[in] Matrix one-norm value. Norm value can be obtained with MatrixNormSyTrid of the original matrix A.

rcond

[out] An estimate of the reciprocal of the condition number. The routine sets rcond=0 if the estimate underflows; in this case the matrix is singular (to working precision). However, anytime rcond is small compared to 1.0, for the working precision, the matrix may be poorly conditioned or even singular.

© 2000–2025, MetaQuotes Ltd.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix L obtained as result of FactorizationLDLSyTridPD method.

The computed *rcond* is never less than r (the reciprocal of the true condition number) and in practice is nearly always less than 10r. A call to this routine involves solving a number of systems of linear equations A*x = b; the number is usually 4 or 5 and never more than 11

© 2000–2025, MetaQuotes Ltd.

CholeskyLinearEquationsSolution

Solves a system of linear equations A * X = B with a real symmetric or complex Hermitian positive-definite matrix using the factorization A = L * L**T computed by FactorizationCholesky, with multiple right-hand sides. LAPACK function POTRS.

Computing for type matrix<double>

```
bool  matrix::CholeskyLinearEquationsSolution(
   matrix&              B,             // right hand side matrix B
   matrix&              X              // solution matrix X
   );

bool  matrix::CholeskyLinearEquationsSolution(
   vector&              B,             // right hand side vector B
   vector&              X              // solution vector X
   );
```

Computing for type matrix<float>

```
bool  matrixf::CholeskyLinearEquationsSolution(
   matrixf&             B,             // right hand side matrix B
   matrixf&             X              // solution matrix X
   );

bool  matrixf::CholeskyLinearEquationsSolution(
   vectorf&             B,             // right hand side vector B
   vectorf&             X              // solution vector X
   );
```

Computing for type matrix<complex>

```
bool  matrixc::CholeskyLinearEquationsSolution(
   matrixc&             B,             // right hand side matrix B
   matrixc&             X              // solution matrix X
   );

bool  matrixc::CholeskyLinearEquationsSolution(
   vectorc&             B,             // right hand side vector B
   vectorc&             X              // solution vector X
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::CholeskyLinearEquationsSolution(
   matrixcf&            B,             // right hand side matrix B
   matrixcf&            X              // solution matrix X
   );

bool  matrixcf::CholeskyLinearEquationsSolution(
   vectorcf&            B,             // right hand side vector B
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&           X             // solution vector X
   );
```

Parameters

B

[in] Matrix B whose columns are the right-hand sides for the systems of equations. Vector B contains one column of right-hand side.

X

[out] Matrix or vector X with solutions of linear equations system.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix L obtained as result of FactorizationCholesky method.

Output matrix X has the same sizes as input matrix B. Output vector X has the same size as input vector B.

© 2000–2025, MetaQuotes Ltd.

CholeskyInverse

Computes the inverse of a real symmetric or complex Hermitian positive-definite matrix using the LLT factorization computed by FactorizationCholesky. LAPACK function POTRI.

Computing for type matrix<double>

```
bool  matrix::CholeskyInverse(
   matrix&         AI             // inverse of matrix A
   );
```

Computing for type matrix<float>

```
bool  matrixf::CholeskyInverse(
   matrixf&        AI             // inverse of matrix A
   );
```

Computing for type matrix<complex>

```
bool  matrixc::CholeskyInverse(
   matrixc&        AI             // inverse of matrix A
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::CholeskyInverse(
   matrixcf&       AI             // inverse of matrix A
   );
```

Parameters

ipiv

[in] Pivot indices array obtained as result of SYTRF or HETRF function.

AI

[out] Inverted matrix.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix L obtained as result of FactorizationCholesky method.

© 2000–2025, MetaQuotes Ltd.

CholeskyCondNumReciprocal

Estimates the reciprocal of the condition number of a real symmetric or complex Hermitian positive-definite matrix A using the LLT factorization computed by FactorizationCholesky. LAPACK function POCON.

Computing for type matrix<double>

```
bool  matrix::CholeskyCondNumReciprocal(
   double           anorm,          // matrix norm value
   double&          rcond           // condition number reciprocal
   );
```

Computing for type matrix<float>

```
bool  matrixf::CholeskyCondNumReciprocal(
   float            anorm,          // matrix norm value
   float&           rcond           // condition number reciprocal
   );
```

Computing for type matrix<complex>

```
bool  matrixc::CholeskyCondNumReciprocal(
   double           anorm,          // matrix norm value
   double&          rcond           // condition number reciprocal
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::CholeskyCondNumReciprocal(
   float            anorm,          // matrix norm value
   float&           rcond           // condition number reciprocal
   );
```

Parameters

anorm

[in] Matrix one-norm value. Norm value can be obtained with MatrixNormSy of the original matrix A.

rcond

[out] An estimate of the reciprocal of the condition number. The routine sets rcond=0 if the estimate underflows; in this case the matrix is singular (to working precision). However, anytime rcond is small compared to 1.0, for the working precision, the matrix may be poorly conditioned or even singular.

Return Value

Return true if successful, otherwise false in case of an error.

Note

This method is applied to the matrix L obtained as result of FactorizationCholesky method.

© 2000–2025, MetaQuotes Ltd.

The computed *rcond* is never less than r (the reciprocal of the true condition number) and in practice is nearly always less than 10r. A call to this routine involves solving a number of systems of linear equations A*x = b; the number is usually 4 or 5 and never more than 11

© 2000–2025, MetaQuotes Ltd.

SylvesterEquationSchur

Solves Sylvester equation for real quasi-triangular or complex triangular matrices:

A*X + X*B = C

where A and B are both upper triangular. A is m-by-m and B is n-by-n; the right hand side C and the solution X are m-by-n.

A and B must be in Schur canonical form (as returned by EigenHessenbergSchurQ or EigenSolverSchur), that is, in case of real quasi-triangular, block upper triangular with 1-by-1 and 2-by-2 diagonal blocks; each 2-by-2 diagonal block has its diagonal elements equal and its off-diagonal elements of opposite sign.

LAPACK function TRSYL.

Computing for type matrix<double>

```
bool  matrix::SylvesterEquationSchur(
   matrix&              B,            // matrix B in Schur form
   matrix&              QA,           // orthogonal matrix of Schur vectors of the inp
   matrix&              QB,           // orthogonal matrix of Schur vectors of the mat
   matrix&              C,            // right hand side matrix C
   matrix&              X             // solution matrix X
   );
```

Computing for type matrix<float>

```
bool  matrixf::SylvesterEquationSchur(
   matrixf&             B,            // matrix B in Schur form
   matrixf&             QA,           // orthogonal matrix of Schur vectors of the inp
   matrixf&             QB,           // orthogonal matrix of Schur vectors of the mat
   matrixf&             C,            // right hand side matrix C
   matrixf&             X             // solution matrix X
   );
```

Computing for type matrix<complex>

```
bool  matrix::SylvesterEquationSchur(
   matrixc&             B,            // matrix B in Schur form
   matrixc&             QA,           // orthogonal matrix of Schur vectors of the inp
   matrixc&             QB,           // orthogonal matrix of Schur vectors of the mat
   matrixc&             C,            // right hand side matrix C
   matrixc&             X             // solution matrix X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SylvesterEquationSchur(
   matrixcf&            B,            // matrix B in Schur form
   matrixcf&            QA,           // orthogonal matrix of Schur vectors of the inp
   matrixcf&            QB,           // orthogonal matrix of Schur vectors of the mat
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixcf&            C,            // right hand side matrix C
   matrixcf&            X             // solution matrix X
   );
```

Parameters

B

[in] Real quasi-triangular or complex triangular matrix B in Schur form (output parameter schur_matrix in EigenSolverSchur or schur_t in EigenHessenbergSchurQ)

QA

[in] Real orthogonal or complex unitary matrix of Schur vectors of the original matrix A (output parameter schur_vectors in EigenSolverSchur or schur_z in EigenHessenbergSchurQ)

QB

[in] Real orthogonal or complex unitary matrix of Schur vectors of the original matrix A (output parameter schur_vectors in EigenSolverSchur or schur_z in EigenHessenbergSchurQ)

C

[in] Matrix C whose columns are the right-hand sides for the systems of equations.

X

[out] Matrix X with solution of Sylvester equation.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix C.

Функция TRSYL обрабатывает только верхнетреугольные матрицы A и B. Для того, чтобы вычислить уравнение Сильвестра для любых квадратных матриц, необходимо предварительно использовать разложение Шура. Например, имеем 2 обычные матрицы A и B, тогда получаем две пары матриц разложения Шура:

A.EigenSolverShur(EIGSCHUR_V,EV,SA,QA);

B.EigenSolverShur(EIGSCHUR_V,EV,SB,QB);

и решаем уравнение Сильвестра:

SA.SylvesterEquationSchur(SB,QA,QB,C,X);

© 2000–2025, MetaQuotes Ltd.

SylvesterEquationSchurBlocked

Solves Sylvester equation for real quasi-triangular or complex triangular matrices:

A*X + X*B = C

where A and B are both upper triangular. A is m-by-m and B is n-by-n; the right hand side C and the solution X are m-by-n.

A and B must be in Schur canonical form (as returned by EigenHessenbergSchurQ or EigenSolverSchur), that is, in case of real quasi-triangular, block upper triangular with 1-by-1 and 2-by-2 diagonal blocks; each 2-by-2 diagonal block has its diagonal elements equal and its off-diagonal elements of opposite sign.

LAPACK function TRSYL3. This is the block (BLAS level 3) version of TRSYL. Faster up to 5 times but not so accurate.

Computing for type matrix<double>

```
bool  matrix::SylvesterEquationSchurBlocked(
   matrix&              B,             // matrix B in Schur form
   matrix&              QA,            // orthogonal matrix of Schur vectors of the inp
   matrix&              QB,            // orthogonal matrix of Schur vectors of the mat
   matrix&              C,             // right hand side matrix C
   matrix&              X              // solution matrix X
   );
```

Computing for type matrix<float>

```
bool  matrixf::SylvesterEquationSchurBlocked(
   matrixf&             B,             // matrix B in Schur form
   matrixf&             QA,            // orthogonal matrix of Schur vectors of the inp
   matrixf&             QB,            // orthogonal matrix of Schur vectors of the mat
   matrixf&             C,             // right hand side matrix C
   matrixf&             X              // solution matrix X
   );
```

Computing for type matrix<complex>

```
bool  matrix::SylvesterEquationSchurBlocked(
   matrixc&             B,             // matrix B in Schur form
   matrixc&             QA,            // orthogonal matrix of Schur vectors of the inp
   matrixc&             QB,            // orthogonal matrix of Schur vectors of the mat
   matrixc&             C,             // right hand side matrix C
   matrixc&             X              // solution matrix X
   );
```

Computing for type matrix<complexf>

```
bool  matrix::SylvesterEquationSchurBlocked(
   matrixcf&            B,             // matrix B in Schur form
   matrixcf&            QA,            // orthogonal matrix of Schur vectors of the inp
```

© 2000–2025, MetaQuotes Ltd.

```
   matrixcf&            QB,           // orthogonal matrix of Schur vectors of the mat
   matrixcf&            C,            // right hand side matrix C
   matrixcf&            X             // solution matrix X
   );
```

Parameters

B

[in] Real quasi-triangular or complex triangular matrix B in Schur form (output parameter schur_matrix in EigenSolverSchur or schur_t in EigenHessenbergSchurQ)

QA

[in] Real orthogonal or complex unitary matrix of Schur vectors of the original matrix A (output parameter schur_vectors in EigenSolverSchur or schur_z in EigenHessenbergSchurQ)

QB

[in] Real orthogonal or complex unitary matrix of Schur vectors of the original matrix A (output parameter schur_vectors in EigenSolverSchur or schur_z in EigenHessenbergSchurQ)

C

[in] Matrix C whose columns are the right-hand sides for the systems of equations.

X

[out] Matrix X with solution of Sylvester equation.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Output matrix X has the same sizes as input matrix C.

Функция TRSYL3 обрабатывает только верхнетреугольные матрицы A и B. Для того, чтобы вычислить уравнение Сильвестра для любых квадратных матриц, необходимо предварительно использовать разложение Шура. Например, имеем 2 обычные матрицы A и B, тогда получаем две пары матриц разложения Шура:

A.EigenSolverShur(EIGSCHUR_V,EV,SA,QA);

B.EigenSolverShur(EIGSCHUR_V,EV,SB,QB);

и решаем уравнение Сильвестра:

SA.SylvesterEquationSchurBlocked(SB,QA,QB,C,X);

© 2000–2025, MetaQuotes Ltd.

There is no special function in the OpenBLAS library to calculate the pseudo-inverse of a matrix. However, for this purpose can be used singular value decomposition (SVD):

1. Decompose matrix A into U * Σ * VT

2. Invert Σ (non-zero singular values only)

3. Get the pseudo-inverse matrix using the formula

$$A+ = V * \Sigma+ * UT$$

Example

```
//+------------------------------------------------------------------+
//|  Тест функции PseudoInverse: проверка A * A+ * A ≈ A             |
//+------------------------------------------------------------------+
int TestPseudoInverse(ulong size_m,ulong size_n)
  {
   matrix matrix_a=matrix::Random(size_m,size_n);
   matrix matrix_pi;
   if(!PseudoInverse(matrix_a,matrix_pi))
      return(1000);

//--- pseudo inverse test  A * A+ * A = A
   matrix matrix_a2=matrix_a @ matrix_pi @ matrix_a;
   int    errors=(int)matrix_a.Compare(matrix_a2,1e-11);

   printf("Pseudo Inverse via SVD %s  matrix size %d x %d  errors=%d",errors==0?"passe

   return(errors);
  }
//+------------------------------------------------------------------+
//| Вычисляет псевдообратную матрицу с помощью сингулярного разлож.  |
//| A+ = V * Σ⁻¹ * Uᵀ                                                |
//+------------------------------------------------------------------+
bool PseudoInverse(matrix& matrix_a,matrix& matrix_pi)
  {
//--- 1. SVD
   matrix matrix_u,matrix_vt;
   vector vector_s;
   if(!matrix_a.SingularValueDecompositionDC(SVDZ_S,vector_s,matrix_u,matrix_vt))
      return(false);

//--- 2. inverse diagonal matrix
   ulong  diag=vector_s.Size();
   matrix matrix_si=matrix::Zeros(diag,diag);
   for(ulong i=0; i<diag; i++)
     {
      if(vector_s[i]!=0.0)
         matrix_si[i][i]=1.0/vector_s[i];
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 3. pseudo inverse  A+ = V * Σ+ * UT
   matrix_pi=matrix_vt.Transpose() @ matrix_si @ matrix_u.Transpose();

   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

There is no special function in the OpenBLAS library to calculate the polar decomposition of a matrix. However, for this purpose can be used singular value decomposition (SVD):

Polar decomposition formula: A = Q * P, where:

Q - orthogonal (or unitary) matrix

P - symmetric (or Hermitian) positive-definite matrix.

How can be calculated polar decomposition using SVD.

SVD formula: A = U * Σ * VT ==> (U * VT) * (V * Σ * VT), ie Q = U * VT and P = V * Σ * VT

Example

```
//+------------------------------------------------------------------+
//| Тест функции PolarDecomposition: проверка Q ортогональна и Q*P=A |
//+------------------------------------------------------------------+
int TestPolarDecomposition(ulong size_m,ulong size_n)
  {
   matrix matrix_a=matrix::Random(size_m,size_n);
   matrix matrix_q,matrix_p;
   if(!PolarDecomposition(matrix_a,matrix_q,matrix_p))
      return(1000);

//--- check orthogonality
   matrix matrix_qqt=(size_m<=size_n) ? matrix_q@matrix_q.Transpose() : matrix_q.Trans
   matrix matrix_i=matrix::Identity(matrix_qqt.Rows(),matrix_qqt.Cols());
   int    errors=(int)matrix_i.Compare(matrix_qqt,1e-11);
   printf("matrix Q is %sorthogonal",errors==0?"":"not ");

//--- polar decomposition test  Q * P = A
   matrix matrix_a2=matrix_q @ matrix_p;
   errors+=(int)matrix_a.Compare(matrix_a2,1e-11);

   printf("Polar Decomposition via SVD %s  matrix size %d x %d  errors=%d",errors==0?'

   return(errors);
  }
//+------------------------------------------------------------------+
//| Полярное разложение матрицы A = Q * P через сингулярное разлож.  |
//| Q — ортогональная, P — положит. определённая симметричная матрица|
//+------------------------------------------------------------------+
bool PolarDecomposition(matrix& matrix_a,matrix& matrix_q,matrix& matrix_p)
  {
//--- SVD
   matrix matrix_u,matrix_vt;
   vector vector_s;
   if(!matrix_a.SingularValueDecompositionDC(SVDZ_S,vector_s,matrix_u,matrix_vt))
      return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- get diagonal matrix
   ulong  diag=vector_s.Size();
   matrix matrix_s=matrix::Zeros(diag,diag);
   matrix_s.Diag(vector_s);

//--- Q = U * VT
   matrix_q=matrix_u @ matrix_vt;
//--- P = V * Σ * VT
   matrix_p=matrix_vt.Transpose() @ matrix_s @ matrix_vt;

   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

Dynamic Mode Decomposition

The Dynamic Mode Decomposition (DMD) functions are designed to compute an evolution operator from data snapshots, enabling efficient analysis of linear dynamics in extended phase spaces. The methods provide memory-efficient representations of evolution operators, with DynamicModeDecompositionQR incorporating QR-based data compression for improved computational efficiency.

Function	Action
DynamicModeDecomposition	Compute the Dynamic Mode Decomposition (DMD) for a pair of data snapshot matrices. LAPACK function GEDMD.
DynamicModeDecompositionQR	Compute the Dynamic Mode Decomposition (DMD) for a pair of data snapshot matrices, using a QR factorization based compression of the data. LAPACK function GEDMDQ.

The Dynamic mode decomposition (DMD) method is an algorithm for searching for an evolution operator (inverse operator problem solutions) in a finite-dimensional problem solution space (numerical or experimentally obtained) in a set of solutions (slices, "snapshots") in some consecutive moments of time. Expansion of the phase space due to the use of a nonlinear basis (relative to the variables of the problem) allows us to construct a global linear operator describing a linear evolution in the extended "rectifying space" (the Coopman operator) and the Perron-Frobenius operator that is its adjoint one. The DMD method is equivalent to a compressed representation of a linear evolution operator in the form of a product of rectangular matrices, which provides significant savings in the required memory during calculations.

© 2000–2025, MetaQuotes Ltd.

DynamicModeDecomposition

Compute the Dynamic Mode Decomposition(по-русски "разложение по динамическим модам") (DMD) for a pair of data snapshot matrices. LAPACK function GEDMD. For the input matrices X and Y such that Y = A*X with an unaccessible matrix A, GEDMD computes a certain number of Ritz pairs of A using the standard Rayleigh-Ritz extraction from a subspace of range(X) that is determined using the leading left singular vectors of X. Optionally, GEDMD returns the residuals of the computed Ritz pairs, the information needed for a refinement of the Ritz vectors, or the eigenvectors of the Exact DMD. Both matrices must be the same size.

Computing for type matrix<double>

```
bool  matrix::DynamicModeDecomposition(
   matrix&               B,                  // second snapshot matrix in the pai
   ENUM_DMD_SCALE        jobs,               // determines whether the initial da
   ENUM_DMD_EIGV         jobz,               // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,               // determines whether to compute the
   ENUM_DMD_REFINE       jobf,               // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,             // allows for a selection of the SVD
   long                  nrnk,               // determines the mode how to comput
   double                tol,                // the tolerance for truncating smal
   vectorc&              eigen_values,       // vector of eigenvalues (Ritz value
   matrix&               left_vectors,       // matrix of left singular vectors
   matrix&               Z                   // matrix of Ritz vectors
   vector&               residuals,          // residuals for the Ritz pairs
   matrix&               res_vectors,        // residual vectors for the Ritz pai
   matrix&               B,                  // computed vectors
   matrix&               W,                  // eigenvectors of the matrix Rayle
   matrix&               S                   // computed vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::DynamicModeDecomposition(
   matrixf&              B,                  // second snapshot matrix in the pai
   ENUM_DMD_SCALE        jobs,               // determines whether the initial da
   ENUM_DMD_EIGV         jobz,               // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,               // determines whether to compute the
   ENUM_DMD_REFINE       jobf,               // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,             // allows for a selection of the SVD
   long                  nrnk,               // determines the mode how to comput
   float                 tol,                // the tolerance for truncating smal
   vectorcf&             eigen_values,       // vector of eigenvalues (Ritz value
   matrixf&              left_vectors,       // matrix of left singular vectors
   matrixf&              Z                   // matrix of Ritz vectors
   vectorf&              residuals,          // residuals for the Ritz pairs
   matrixf&              res_vectors,        // residual vectors for the Ritz pai
   matrixf&              B,                  // computed vectors
   matrixf&              W,                  // eigenvectors of the matrix Rayle
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixf&              S                       // computed vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::DynamicModeDecomposition(
   matrixc&              B,                      // second snapshot matrix in the pa
   ENUM_DMD_SCALE        jobs,                   // determines whether the initial da
   ENUM_DMD_EIGV         jobz,                   // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,                   // determines whether to compute the
   ENUM_DMD_REFINE       jobf,                   // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,                 // allows for a selection of the SVD
   long                  nrnk,                   // determines the mode how to comput
   double                tol,                    // the tolerance for truncating smal
   vectorc&              eigen_values,           // vector of eigenvalues (Ritz value
   matrixc&              left_vectors,           // matrix of left singular vectors
   matrixc&              Z                       // matrix of Ritz vectors
   vector&               residuals,              // residuals for the Ritz pairs
   matrixc&              res_vectors,            // residual vectors for the Ritz pa
   matrixc&              B,                      // computed vectors
   matrixc&              W,                      // eigenvectors of the matrix Rayle
   matrixc&              S                       // computed vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::DynamicModeDecomposition(
   matrixcf&             B,                      // second snapshot matrix in the pa
   ENUM_DMD_SCALE        jobs,                   // determines whether the initial da
   ENUM_DMD_EIGV         jobz,                   // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,                   // determines whether to compute the
   ENUM_DMD_REFINE       jobf,                   // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,                 // allows for a selection of the SVD
   long                  nrnk,                   // determines the mode how to comput
   float                 tol,                    // the tolerance for truncating smal
   vectorcf&             eigen_values,           // vector of eigenvalues (Ritz value
   matrixcf&             left_vectors,           // matrix of left singular vectors
   matrixcf&             Z                       // matrix of Ritz vectors
   vectorf&              residuals,              // residuals for the Ritz pairs
   matrixcf&             res_vectors,            // residual vectors for the Ritz pa
   matrixcf&             B,                      // computed vectors
   matrixcf&             W,                      // eigenvectors of the matrix Rayle
   matrixcf&             S                       // computed vectors
   );
```

Parameters

Y

[in] The second snapshot matrix in the pair.

© 2000-2025, MetaQuotes Ltd.

jobs

[in] Value from the ENUM_DMD_SCALE enumeration which determines whether the initial data snapshots are scaled by a diagonal matrix.

jobz

[in] Value from the ENUM_DMD_EIGV enumeration which determines whether the eigenvectors (Koopman modes) will be computed.

jobr

[in] Value from the ENUM_DMD_RESIDUALS enumeration which determines whether to compute the residuals.

jobf

[in] Value from the ENUM_DMD_REFINE enumeration which specifies whether to store information needed for post-processing (e.g. computing refined Ritz vectors).

whtsvd

[in] Value from the ENUM_SVD_ALG enumeration which allows for a selection of the SVD algorithm from the LAPACK library.

nrnk

[in] Value determines the mode how to compute the numerical rank, i.e. how to truncate small singular values of the input matrix X. If

NRNK = -1 :: i-th singular value sigma(i) is truncated if sigma(i) <= TOL*sigma(1). This option is recommended.

NRNK = -2 :: i-th singular value sigma(i) is truncated if sigma(i) <= TOL*sigma(i-1). This option is included for R&D purposes. It requires highly accurate SVD, which may not be feasible.

The numerical rank can be enforced by using positive value of NRNK as follows: 0 < NRNK <= N :: at most NRNK largest singular values will be used. If the number of the computed nonzero singular values is less than NRNK, then only those nonzero values will be used and the actually used dimension is less than NRNK.

tol

[in] The tolerance for truncating small singular values. 0 <= TOL < 1

eigen_values

[out] Vector of eigenvalues of size K. The leading K (K<=N) entries of EIGS contain the computed eigenvalues (Ritz values).

left_vectors

[out] Matrix of left singular vectors. The leading K columns contain a POD basis (POD - Proper Orthogonal Decomposition, правильная ортогональная декомпозиция), i.e. the leading K left singular vectors of the input data matrix X All N columns of contain all left singular vectors of the input matrix X.

Z

[out] Matrix of Ritz vectors. If JOBZ =='V' then Z contains the Ritz vectors. Z(:,i) is an eigenvector of the i-th Ritz value; ||Z(:,i)||_2=1. If JOBZ == 'F', then the Z(:,i)'s are given implicitly as the columns of X(:,1:K)*W(1:K,1:K), i.e. X(:,1:K)*W(:,i) is an eigenvector corresponding to EIGS(i). The columns of W(1:k,1:K) are the computed eigenvectors of the K-by-K Rayleigh quotient.

© 2000–2025, MetaQuotes Ltd.

residuals

[out] Residuals for the K computed Ritz pairs.

res_vectors

[out] Residual vectors for the K computed Ritz pairs.

B

[out] M-by-K matrix (K<=N).

If JOBF =='R', B(1:M,1:K) contains A*U(:,1:K), and can be used for computing the refined vectors; see further details in the provided references.

If JOBF == 'E', B(1:M,1:K) contains A*U(:,1:K)*W(1:K,1:K), which are the vectors from the Exact DMD, up to scaling by the inverse eigenvalues.

If JOBF =='N', then B is not referenced.

W

[out] K-by-K matrix (K<=N). Contains the K computed eigenvectors of the matrix Rayleigh quotient. The Ritz vectors (returned in Z) are the product of X (containing a POD basis for the input matrix X) and W.

S

[out] K-by-K matrix (K<=N). The array S(1:K,1:K) is used for the matrix Rayleigh quotient. This content is overwritten during the eigenvalue decomposition by GEEV.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The number of matrices rows (M) must not be less than the number of columns (N).

ENUM_DMD_SCALE

An enumeration that determines whether the initial data snapshots are scaled by a diagonal matrix.

ID	Description
DMDSCALE_S	'S': The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that X*D has unit nonzero columns
DMDSCALE_C	'C': The snapshots are scaled as with the 'S' option.If it is found that an i-th column of X is zero vector and the corresponding i-th column of Y is non-zero, then the i-th column of Y is set to zero
DMDSCALE_Y	'Y': The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that Y*D has unit nonzero columns
DMDSCALE_N	'N': No data scaling

© 2000–2025, MetaQuotes Ltd.

ENUM_DMD_EIGV

An enumeration which determines whether the eigenvectors (Koopman modes) will be computed.

ID	Description
DMDEIGV_V	'V': The eigenvectors (Koopman modes) will be computed
DMDEIGV_F	'F': The eigenvectors (Koopman modes) will be returned in factored form
DMDEIGV_N	'N': The eigenvectors are not computed

ENUM_DMD_RESIDUALS

An enumeration which determines whether to compute the residuals.

ID	Description
DMDRESIDUALS_R	'R': The residuals for the computed eigenpairs will be computed
DMDRESIDUALS_N	'N': The residuals are not computed

ENUM_DMD_REFINE

An enumeration which specifies whether to store information needed for post-processing.

ID	Description
DMDREFINE_R	'R': The matrix needed for the refinement of the Ritz vectors is computed and stored in the array B
DMDREFINE_E	'E': The unscaled eigenvectors of the Exact DMD are computed and returned in the array B
DMDREFINE_N	'N': No eigenvector refinement data is computed

ENUM_SVD_ALG

An enumeration selecting the SVD algorythm.

ID	Description
SVDALG_1	1: GESVD (the QR SVD algorithm)
SVDALG_2	2: GESDD (the Divide and Conquer algorithm)

© 2000–2025, MetaQuotes Ltd.

ID	Description
SVDALG_3	3: GESVDQ (the preconditioned QR SVD; this and 4 are the most accurate options)
SVDALG_4	4: GEJSV (the preconditioned Jacobi SVD; this and 3 are the most accurate options)

© 2000-2025, MetaQuotes Ltd.

DynamicModeDecompositionQR

Compute the Dynamic Mode Decomposition(по-русски "декомпозиция динамических мод") (DMD) for a pair of data snapshot matrices, using a QR factorization based compression of the data. LAPACK function GEDMDQ. For the input matrices X and Y such that Y = A*X with an unaccessible matrix A, GEDMDQ computes a certain number of Ritz pairs of A using the standard Rayleigh-Ritz extraction from a subspace of range(X) that is determined using the leading left singular vectors of X. Optionally, GEDMDQ returns the residuals of the computed Ritz pairs, the information needed for a refinement of the Ritz vectors, or the eigenvectors of the Exact DMD.

The input M-by-N matrix F. The columns of F are the sequence of data snapshots from a single trajectory, taken at equidistant discrete times. It is assumed that the column norms of F are in the range of the normalized floating point numbers.

Computing for type matrix<double>

```
bool  matrix::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE        jobs,                 // determines whether the initial da
   ENUM_DMDQ_EIGV        jobz,                 // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,                 // determines whether to compute the
   ENUM_DMDQ_Q           jobq,                 // specifies whether to explicitly c
   ENUM_DMDQ_R           jobt,                 // specifies whether to return the u
   ENUM_DMD_REFINE       jobf,                 // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,               // allows for a selection of the SVD
   long                  nrnk,                 // determines the mode how to comput
   double                tol,                  // the tolerance for truncating smal
   vectorc&              eigen_values,         // vector of eigenvalues (Ritz value
   matrix&               Q,                    // Q factor from the QR factorizatio
   matrix&               R,                    // R factor from the QR factorizatio
   matrix&               left_vectors,         // matrix of left singular vectors
   matrix&               Z                     // matrix of Ritz vectors
   vector&               residuals,            // residuals for the Ritz pairs
   matrix&               B,                    // computed vectors
   matrix&               V,                    // eigenvectors of the matrix Rayleig
   matrix&               S                     // computed vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE        jobs,                 // determines whether the initial da
   ENUM_DMDQ_EIGV        jobz,                 // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS    jobr,                 // determines whether to compute the
   ENUM_DMDQ_Q           jobq,                 // specifies whether to explicitly c
   ENUM_DMDQ_R           jobt,                 // specifies whether to return the u
   ENUM_DMD_REFINE       jobf,                 // specifies whether to store inform
   ENUM_SVD_ALG          whtsvd,               // allows for a selection of the SVD
   long                  nrnk,                 // determines the mode how to comput
   float                 tol,                  // the tolerance for truncating smal
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&            eigen_values,          // vector of eigenvalues (Ritz value
   matrixf&             Q,                     // Q factor from the QR factorizatio
   matrixf&             R,                     // R factor from the QR factorizatio
   matrixf&             left_vectors,          // matrix of left singular vectors
   matrixf&             Z                      // matrix of Ritz vectors
   vectorf&             residuals,             // residuals for the Ritz pairs
   matrixf&             B,                     // computed vectors
   matrixf&             V,                     // eigenvectors of the matrix Rayle
   matrixf&             S                      // computed vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE       jobs,                  // determines whether the initial da
   ENUM_DMDQ_EIGV       jobz,                  // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS   jobr,                  // determines whether to compute the
   ENUM_DMDQ_Q          jobq,                  // specifies whether to explicitly c
   ENUM_DMDQ_R          jobt,                  // specifies whether to return the u
   ENUM_DMD_REFINE      jobf,                  // specifies whether to store inform
   ENUM_SVD_ALG         whtsvd,                // allows for a selection of the SVD
   long                 nrnk,                  // determines the mode how to comput
   double               tol,                   // the tolerance for truncating smal
   vectorc&             eigen_values,          // vector of eigenvalues (Ritz value
   matrixc&             Q,                     // Q factor from the QR factorizatio
   matrixc&             R,                     // R factor from the QR factorizatio
   matrixc&             left_vectors,          // matrix of left singular vectors
   matrixc&             Z                      // matrix of Ritz vectors
   vector&              residuals,             // residuals for the Ritz pairs
   matrixc&             B,                     // computed vectors
   matrixc&             V,                     // eigenvectors of the matrix Rayle
   matrixc&             S                      // computed vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::DynamicModeDecompositionQR(
   ENUM_DMD_SCALE       jobs,                  // determines whether the initial da
   ENUM_DMDQ_EIGV       jobz,                  // determines whether the eigenvecto
   ENUM_DMD_RESIDUALS   jobr,                  // determines whether to compute the
   ENUM_DMDQ_Q          jobq,                  // specifies whether to explicitly c
   ENUM_DMDQ_R          jobt,                  // specifies whether to return the u
   ENUM_DMD_REFINE      jobf,                  // specifies whether to store inform
   ENUM_SVD_ALG         whtsvd,                // allows for a selection of the SVD
   long                 nrnk,                  // determines the mode how to comput
   float                tol,                   // the tolerance for truncating smal
   vectorcf&            eigen_values,          // vector of eigenvalues (Ritz value
   matrixcf&            Q,                     // Q factor from the QR factorizatio
```

© 2000-2025, MetaQuotes Ltd.

```
   matrixcf&            R,                  // R factor from the QR factorizatio
   matrixcf&            left_vectors,       // matrix of left singular vectors
   matrixcf&            Z                   // matrix of Ritz vectors
   vectorf&             residuals,          // residuals for the Ritz pairs
   matrixcf&            B,                  // computed vectors
   matrixcf&            V,                  // eigenvectors of the matrix Rayle
   matrixcf&            S                   // computed vectors
   );
```

Parameters

jobs

[in] Value from the ENUM_DMD_SCALE enumeration which determines whether the initial data snapshots are scaled by a diagonal matrix. The data snapshots are the columns of F. The leading N-1 columns of F are denoted X and the trailing N-1 columns are denoted Y.

jobz

[in] Value from the ENUM_DMDQ_EIGV enumeration which determines whether the eigenvectors (Koopman modes) will be computed.

jobr

[in] Value from the ENUM_DMD_RESIDUALS enumeration which determines whether to compute the residuals.

jobq

[in] Value from the ENUM_DMDQ_Q enumeration which specifies whether to explicitly compute and return the unitary matrix from the QR factorization of the data snapshot matrix.

jobt

[in] Value from the ENUM_DMDQ_R enumeration which specifies whether to return the upper triangular factor from the QR factorization of the data snapshot matrix.

jobf

[in] Value from the ENUM_DMD_REFINE enumeration which specifies whether to store information needed for post-processing (e.g. computing refined Ritz vectors).

whtsvd

[in] Value from the ENUM_SVD_ALG enumeration which allows for a selection of the SVD algorithm from the LAPACK library.

nrnk

[in] Value determines the mode how to compute the numerical rank, i.e. how to truncate small singular values of the input matrix X. If

NRNK = -1 :: i-th singular value sigma(i) is truncated if sigma(i) <= TOL*sigma(1). This option is recommended.

NRNK = -2 :: i-th singular value sigma(i) is truncated if sigma(i) <= TOL*sigma(i-1). This option is included for R&D purposes. It requires highly accurate SVD, which may not be feasible.

The numerical rank can be enforced by using positive value of NRNK as follows: 0 < NRNK <= N-1 :: at most NRNK largest singular values will be used. If the number of the computed nonzero

© 2000–2025, MetaQuotes Ltd.

singular values is less than NRNK, then only those nonzero values will be used and the actually used dimension is less than NRNK.

tol

[in] The tolerance for truncating small singular values. 0 <= TOL < 1

eigen_values

[out] Vector of eigenvalues of size K. The leading K (K<N) entries of EIGS contain the computed eigenvalues (Ritz values).

Q

[out] Orthogonal matrix/factor of the QR factorization of the initial data snapshots matrix F.

R

[out] N-by-N upper triangular factor from the QR factorization of the data snapshot matrix F.

left_vectors

[out] Matrix of left singular vectors. The leading K columns of X contain the leading K left singular vectors of the hold representations of the leading N-1 snapshots in the orthonormal basis computed in the QR factorization of F. To lift them to the space of the left singular vectors U(:,1:K) of the input data, pre-multiply with the Q factor from the initial QR factorization.

Z

[out] M-by-K matrix

If JOBZ =='V' then Z contains the Ritz vectors. Z(:,i) is an eigenvector of the i-th Ritz value; ||Z(:,i)||_2=1.

If JOBZ == 'F', then the Z(:,i)'s are given implicitly as Z*V, where Z contains orthonormal matrix (the product of Q from the initial QR factorization and the SVD/POD_basis returned by GEDMD in X) and the second factor (the eigenvectors of the Rayleigh quotient) is in the matrix V, as returned by GEDMD. That is, X(:,1:K)*V(:,i) is an eigenvector corresponding to EIGS(i). The columns of V(1:K,1:K) are the computed eigenvectors of the K-by-K Rayleigh quotient.

residuals

[out] Residuals for the K computed Ritz pairs.

B

[out] N-by-K matrix (K<N).

If JOBF =='R', B(1:N,1:K) contains A*U(:,1:K), and can be used for computing the refined vectors.

If JOBF == 'E', B(1:N,1;K) contains A*U(:,1:K)*W(1:K,1:K), which are the vectors from the Exact DMD, up to scaling by the inverse eigenvalues.

In both cases, the content of B can be lifted to the original dimension of the input data by pre-multiplying with the Q factor from the initial QR factorization. Here A denotes a compression of the underlying operator.

If JOBF =='N', then B is not referenced.

V

[out] K-by-K matrix (K<N). Contains the the K eigenvectors of the Rayleigh quotient. The Ritz vectors (returned in Z) are the product of Q from the initial QR factorization.

© 2000–2025, MetaQuotes Ltd.

S

[out] K-by-K matrix (K<N). The array S(1:K,1:K) is used for the matrix Rayleigh quotient. This content is overwritten during the eigenvalue decomposition by GEEV.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The number of matrices rows (M) must not be less than the number of columns (N).

ENUM_DMD_SCALE

An enumeration that determines whether the initial data snapshots are scaled by a diagonal matrix.

ID	Description
DMDSCALE_S	'S': The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that X*D has unit nonzero columns
DMDSCALE_C	'C': The snapshots are scaled as with the 'S' option.If it is found that an i-th column of X is zero vector and the corresponding i-th column of Y is non-zero, then the i-th column of Y is set to zero
DMDSCALE_Y	'Y': The data snapshots matrices X and Y are multiplied with a diagonal matrix D so that Y*D has unit nonzero columns
DMDSCALE_N	'N': No data scaling

ENUM_DMDQ_EIGV

An enumeration which determines whether the eigenvectors (Koopman modes) will be computed.

ID	Description
DMDQEIGV_V	'V': The eigenvectors (Koopman modes) will be computed
DMDQEIGV_F	'F': The eigenvectors (Koopman modes) will be returned in factored form as the product Z*V
DMDQEIGV_Q	'Q': The eigenvectors (Koopman modes) will be returned in factored form as the product Q*Z
DMDQEIGV_N	'N': The eigenvectors are not computed

ENUM_DMD_RESIDUALS

© 2000–2025, MetaQuotes Ltd.

An enumeration which determines whether to compute the residuals.

ID	Description
DMDRESIDUALS_R	'R': The residuals for the computed eigenpairs will be computed
DMDRESIDUALS_N	'N': The residuals are not computed

ENUM_DMDQ_Q

An enumeration which specifies whether to explicitly compute and return the unitary matrix from the QR factorization.

ID	Description
DMDQQ_Q	'Q': The matrix Q of the QR factorization of the data snapshot matrix is computed
DMDQQ_N	'N': The matrix Q is not explicitly computed

ENUM_DMDQ_R

An enumeration whichspecifies whether to return the upper triangular factor from the QR factorization.

ID	Description
DMDQR_R	'R': The matrix R of the QR factorization of the data snapshot matrix is computed
DMDQR_N	'N': The matrix R is not explicitly computed

ENUM_DMD_REFINE

An enumeration which specifies whether to store information needed for post-processing.

ID	Description
DMDREFINE_R	'R': The matrix needed for the refinement of the Ritz vectors is computed and stored in the array B
DMDREFINE_E	'E': The unscaled eigenvectors of the Exact DMD are computed and returned in the array B
DMDREFINE_N	'N': No eigenvector refinement data is computed

© 2000–2025, MetaQuotes Ltd.

ENUM_SVD_ALG

An enumeration selecting the SVD algorythm.

ID	Description
SVDALG_1	1: GESVD (the QR SVD algorithm)
SVDALG_2	2: GESDD (the Divide and Conquer algorithm)
SVDALG_3	3: GESVDQ (the preconditioned QR SVD; this and 4 are the most accurate options)
SVDALG_4	4: GEJSV (the preconditioned Jacobi SVD; this and 3 are the most accurate options)

© 2000–2025, MetaQuotes Ltd.

Singular Spectrum Analysis

The section contains functions for decomposing a matrix into three components: orthogonal matrices and a diagonal matrix of singular values. SVD is applied to solve various linear algebra problems such as data dimensionality reduction, image compression, solving systems of equations, as well as data analysis and optimization. The main functions allow calculating singular values and vectors, reconstruct matrices, as well as approximate them with reduced rank accuracy.

Function	Action
SingularSpectrumAnalysisSpectrum	A method function for calculating the relative contributions of spectral components based on their eigenvalues.
SingularSpectrumAnalysisForecast	A method function for calculating reconstructed and predicted data using spectral components of the input time series.
SingularSpectrumAnalysisReconstructComponents	A method function for calculating reconstructed components of the input time series and their contributions.
SingularSpectrumAnalysisReconstructSeries	A method function for calculating the reconstructed time series using the first component_count components.

© 2000–2025, MetaQuotes Ltd.

SingularSpectrumAnalysisSpectrum

A method function for calculating the relative contributions of spectral components based on their eigenvalues.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisSpectrum(
   ulong   window_length,      // window size for constructing the trajectory matrix
   vector& spectrum            // vector of component contributions to the input serie
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisSpectrum(
   ulong   window_length,      // window size for constructing the trajectory matrix
   vectorc& spectrum           // vector of component contributions to the input serie
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

spectrum

[out] Vector of component contributions to the input series - eigenvalues of the covariance matrix of the input time series.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000-2025, MetaQuotes Ltd.

SingularSpectrumAnalysisForecast

A method function for calculating reconstructed and predicted data using spectral components of the input time series.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisForecast(
   ulong    window_length,         // window size for constructing the trajectory mati
   ulong    component_count,       // number of components used for forecasting
   ulong    forecast_horizon,      // number of points to forecast
   vector&  forecast               // vector consisting of reconstructed and predicted
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisForecast(
   ulong    window_length,         // window size for constructing the trajectory mati
   ulong    component_count,       // number of components used for forecasting
   ulong    forecast_horizon,      // number of points to forecast
   vectorc& forecast               // vector consisting of reconstructed and predicted
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

component_count

[in] Number of components used for forecasting.

forecast_horizon

[in] Number of points to forecast.

forecast

[out] Combining points reconstructed by *component_count* plus *forecast_horizon* points predicted using the first *component_count* components. Thus, the *forecast* vector has the size of (T+forecast_horizon), where T is the input series length.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000–2025, MetaQuotes Ltd.

SingularSpectrumAnalysisReconstructComponents

A method function for calculating reconstructed components of the input time series and their contributions.

Calculations for vector<double> and matrix<double> types

```
bool  vector::SingularSpectrumAnalysisReconstructComponents(
   ulong    window_length,      // window size for constructing the trajectory matrix
   matrix&  components,         // matrix of reconstructed components
   vector&  contributions       // vector of component contributions to the input seri
   );
```

Calculations for vector<complex> and matrix<complex> types

```
bool  vector::SingularSpectrumAnalysisReconstructComponents(
   ulong    window_length,      // window size for constructing the trajectory matrix
   matrixc& components,         // matrix of reconstructed components
   vectorc& contributions       // vector of component contributions to the input seri
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

components

[out] A matrix of reconstructed components, where each column describes a separate component.

contributions

[out] Vector of component contributions to the input series (eigenvalues of the covariance matrix of the input time series).

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000–2025, MetaQuotes Ltd.

SingularSpectrumAnalysisReconstructSeries

A method function for calculating the reconstructed time series using the first *component_count* components.

Calculations for vector<double> type

```
bool  vector::SingularSpectrumAnalysisReconstructSeries(
   ulong    window_length,       // window size for constructing the trajectory matri
   ulong    component_count,     // number of components used for reconstruction
   vector&  reconstructed        // reconstructed time series
   );
```

Calculations for vector<complex> type

```
bool  vector::SingularSpectrumAnalysisReconstructSeries(
   ulong    window_length,       // window size for constructing the trajectory matri
   ulong    component_count,     // number of components used for reconstruction
   vectorc& reconstructed        // reconstructed time series
   );
```

Parameters

window_length

[in] Window size for constructing the trajectory matrix, the number of components the input time series should be decomposed into.

component_count

[out] Number of components used for time series reconstruction.

reconstructed

[out] Vector containing the reconstructed output series.

Return Value

The function returns 'true' on success or 'false' if an error occurs.

Note

The *window_length* parameter value should be less than the size of the input time series. To construct a full-fledged trajectory matrix, the optimal size is considered to be approximately equal to half the size of the input time series.

© 2000–2025, MetaQuotes Ltd.

Reductions

Functions designed to transform matrices into special forms using orthogonal transformations. These methods are widely used in numerical computations, such as spectral analysis, eigenvalue problems, and singular value decomposition (SVD) of matrices.

These functions enable efficient matrix transformations essential for solving linear algebra problems, including computing Singular Value Decomposition (SVD) and finding eigenvalues of symmetric matrices. The use of LAPACK functions ensures high performance and computational accuracy.

Function	Action
ReduceToBidiagonal	Reduces a general real or complex m-by-n matrix A to upper or lower bidiagonal form B by an orthogonal transformation: Q**T * A * P = B. If m >= n, B is upper bidiagonal; if m < n, B is lower bidiagonal. LAPACK function GEBRD.
ReflectBidiagonalToQP	Generates orthogonal matrices Q and P**T (or P**H for complex types) determined by ReduceToBidiagonal method when reducing a real or complex matrix A to bidiagonal form: A = Q * B * P**T. Q and P**T are defined as products of elementary reflectors H(i) or G(i) respectively. LAPACK functions ORGBR, UNGBR.
ReduceSymmetricToTridiagonal	Reduces a real symmetric or complex Hermitian matrix A to trdiagonal form B by an orthogonal similarity transformation: Q**T * A * Q = B. LAPACK functions SYTRD, HETRD.
ReflectTridiagonalToQ	Generates orthogonal matrix Q which is defined as the product of n-1 elementary reflectors of order n, as returned by ReduceSymmetricToTridiagonal
ReduceToHessenberg	Reduces a real or complex general n-by-n matrix A to upper Hessenberg form B by an orthogonal similarity transformation: Q**T * A * Q = H. LAPACK function GEHRD.
ReflectHessenbergToQ	Generates orthogonal matrix Q which is defined as the product of n-1 elementary reflectors of order n, as returned by ReduceToHessenberg: Q = H(1) H(2) . . . H(n-1). LAPACK function ORGHR.

© 2000–2025, MetaQuotes Ltd.

ReduceToBidiagonal

Reduces a general real or complex m-by-n matrix A to upper or lower bidiagonal form B by an orthogonal transformation: Q**T * A * P = B. If m >= n, B is upper bidiagonal; if m < n, B is lower bidiagonal. LAPACK function GEBRD.

Computing for type matrix<double>

```
bool  matrix::ReduceToBidiagonal(
   matrix&          B,             // bidiagonal matrix
   matrix&          reflect_qp,    // q and p-reflectors
   vector&          tau_q,         // scalar factors of the elementary reflectors Q
   vector&          tau_p          // scalar factors of the elementary reflectors P
   );
```

Computing for type matrix<float>

```
bool  matrix::ReduceToBidiagonal(
   matrixf&         B,             // bidiagonal matrix
   matrixf&         reflect_qp,    // q and p-reflectors
   vectorf&         tau_q,         // scalar factors of the elementary reflectors Q
   vectorf&         tau_p          // scalar factors of the elementary reflectors P
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReduceToBidiagonal(
   matrixc&         B,             // bidiagonal matrix
   matrixc&         reflect_qp,    // q and p-reflectors
   vectorc&         tau_q,         // scalar factors of the elementary reflectors Q
   vectorc&         tau_p          // scalar factors of the elementary reflectors P
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReduceToBidiagonal(
   matrixcf&        B,             // bidiagonal matrix
   matrixcf&        reflect_qp,    // q and p-reflectors
   vectorcf&        tau_q,         // scalar factors of the elementary reflectors Q
   vectorcf&        tau_p          // scalar factors of the elementary reflectors P
   );
```

Parameters

B

[out] Upper or lower bidiagonal matrix.

reflect_qp

[out] Transformed matrix A. If m >= n, the diagonal and the first superdiagonal are overwritten with the upper bidiagonal matrix B; the elements below the diagonal, with the vector tau_q, represent the orthogonal matrix Q as a product of elementary reflectors, and the elements above the first superdiagonal, with the vector tau_p, represent the orthogonal matrix P as a product of

© 2000–2025, MetaQuotes Ltd.

elementary reflectors; if m < n, the diagonal and the first subdiagonal are overwritten with the lower bidiagonal matrix B; the elements below the first subdiagonal, with the vector tau_q, represent the orthogonal matrix Q as a product of elementary reflectors, and the elements above the diagonal, with the vector tau_p, represent the orthogonal matrix P as a product of elementary reflectors.

tau_q

[out] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

tau_p

[out] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix P.

Return Value

Return true if successful, otherwise false in case of an error.

Note

The matrices Q and P are represented as products of elementary reflectors:

If m >= n,

Q = H(1) H(2) . . . H(n) and P = G(1) G(2) . . . G(n-1)

Each H(i) and G(i) has the form:

H(i) = I - tauq * v * v**T and G(i) = I - taup * u * u**T

where tauq and taup are scalars, and v and u are vectors;

v(1:i-1) = 0, v(i) = 1, and v(i+1:m) is stored on exit in A(i+1:m,i);

u(1:i) = 0, u(i+1) = 1, and u(i+2:n) is stored on exit in A(i,i+2:n);

tauq is stored in *tau_q*(i) and taup in *tau_p*(i).

If m < n,

Q = H(1) H(2) . . . H(m-1) and P = G(1) G(2) . . . G(m)

Each H(i) and G(i) has the form:

H(i) = I - tauq * v * v**T and G(i) = I - taup * u * u**T

where tauq and taup are scalars, and v and u are vectors;

v(1:i) = 0, v(i+1) = 1, and v(i+2:m) is stored on exit in A(i+2:m,i);

u(1:i-1) = 0, u(i) = 1, and u(i+1:n) is stored on exit in A(i,i+1:n);

tauq is stored in *tau_q*(i) and taup in *tau_p*(i).

The contents of A on exit (output matrix *reflect_qp*) are illustrated by the following examples:

© 2000–2025, MetaQuotes Ltd.

```
m = 6 and n = 5 (m > n):          m = 5 and n = 6 (m < n):
   (  d   e   u1  u1  u1 )           (  d   u1  u1  u1  u1  u1 )
   (  v1  d   e   u2  u2 )           (  e   d   u2  u2  u2  u2 )
   (  v1  v2  d   e   u3 )           (  v1  e   d   u3  u3  u3 )
   (  v1  v2  v3  d   e  )           (  v1  v2  e   d   u4  u4 )
   (  v1  v2  v3  v4  d  )           (  v1  v2  v3  e   d   u5 )
   (  v1  v2  v3  v4  v5 )
```

where d and e denote diagonal and off-diagonal elements of B, vi denotes an element of the vector defining H(i), and ui an element of the vector defining G(i).

Matrices Q and P can be produced with ReflectBidiagonalToQP method.

© 2000–2025, MetaQuotes Ltd.

ReflectBidiagonalToQP

Generates orthogonal matrices Q and P**T (or P**H for complex types) determined by ReduceToBidiagonal method when reducing a real or complex matrix A to bidiagonal form: A = Q * B * P**T. Q and P**T are defined as products of elementary reflectors H(i) or G(i) respectively. LAPACK functions ORGBR, UNGBR.

As input is used transformed matrix reflect_qp with the same sizes m-by-n as in original matrix A.

Computing for type matrix<double>

```
bool  matrix::ReflectBidiagonalToQP(
   vector&          tau_q,         // scalar factors of the elementary reflectors Q
   vector&          tau_p,         // scalar factors of the elementary reflectors P
   matrix&          Q,             // matrix Q
   matrix&          PT             // transposed matrix P
   );
```

Computing for type matrix<float>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorf&         tau_q,         // scalar factors of the elementary reflectors Q
   vectorf&         tau_p,         // scalar factors of the elementary reflectors P
   matrixf&         Q,             // matrix Q
   matrixf&         PT             // transposed matrix P
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorc&         tau_q,         // scalar factors of the elementary reflectors Q
   vectorc&         tau_p,         // scalar factors of the elementary reflectors P
   matrixc&         Q,             // matrix Q
   matrixc&         PH             // hermitian conjugated matrix P
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReflectBidiagonalToQP(
   vectorcf&        tau_q,         // scalar factors of the elementary reflectors Q
   vectorcf&        tau_p,         // scalar factors of the elementary reflectors P
   matrixcf&        Q,             // matrix Q
   matrixcf&        PH             // hermitian conjugated matrix P
   );
```

Parameters

tau_q

[in] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

tau_p

© 2000–2025, MetaQuotes Ltd.

[in] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix P.

Q

[out] Orthogonal matrix Q.

PT (PH)

[out] Transposed (or hermitian conjugated) matrix P.

Return Value

Return true if successful, otherwise false in case of an error.

Note

If m >= n, matrix Q is of m-by-n sizes, matrix PT is of n-by-n sizes.

If m < n, matrix Q is of m-by-m sizes, matrix PT is of m-by-n sizes.

© 2000–2025, MetaQuotes Ltd.

ReduceSymmetricToTridiagonal

Reduces a real symmetric or complex Hermitian matrix A to trdiagonal form B by an orthogonal similarity transformation: Q**T * A * Q = B. LAPACK functions SYTRD, HETRD.

Computing for type matrix<double>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrix&         B,             // tridiagonal matrix
   matrix&         reflect_q,     // q-reflectors
   vector&         tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<float>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixf&        B,             // tridiagonal matrix
   matrixf&        reflect_q,     // q-reflectors
   vectorf&        tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixc&        B,             // tridiagonal matrix
   matrixc&        reflect_q,     // q-reflectors
   vectorc&        tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReduceSymmetricToTridiagonal(
   matrixcf&       B,             // tridiagonal matrix
   matrixcf&       reflect_q,     // q and p-reflectors
   vectorcf&       tau_q          // scalar factors of the elementary reflectors Q
   );
```

Parameters

B

[out] Symmetric (or Hermitian) tridiagonal matrix.

reflect_q

[out] Upper or lower triangular matrix, it depends on the input matrix A. In upper case the diagonal and first superdiagonal of A are overwritten by the corresponding elements of the tridiagonal matrix B, and the elements above the first superdiagonal, with the array *tau_q*, represent the orthogonal matrix Q as a product of elementary reflectors; in lower case the diagonal and first subdiagonal of A are overwritten by the corresponding elements of the tridiagonal matrix B, and the elements below the first subdiagonal, with the array *tau_q*, represent the orthogonal matrix Q as a product of elementary reflectors.

tau_q

© 2000–2025, MetaQuotes Ltd.

[out] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

Note

In upper case the matrix Q is represented as a product of elementary reflectors

Q = H(n-1) . . . H(2) H(1)

Each H(i) has the form

H(i) = I - tau * v * v**T

where tau is a scalar, and v is a vector with v(i+1:n) = 0 and v(i) = 1; v(1:i-1) is stored on exit in A(1:i-1,i+1), and tau in *tau_q*(i).

In lower case the matrix Q is represented as a product of elementary reflectors

Q = H(1) H(2) . . . H(n-1)

Each H(i) has the form

H(i) = I - tau * v * v**T

where tau is a scalar, and v is a vector with v(1:i) = 0 and v(i+1) = 1; v(i+2:n) is stored on exit in A(i+2:n,i), and tau in *tau_q*(i). */

The contents of A on exit (output matrix *reflect_q*) are illustrated by the following examples with n = 5:

```
      upper case:                          lower case:

      (  d   e   v2  v3  v4 )              (  d                   )
      (      d   e   v3  v4 )              (  e   d               )
      (          d   e   v4 )              (  v1  e   d           )
      (              d   e  )              (  v1  v2  e   d       )
      (                  d  )              (  v1  v2  v3  e   d   )
```

where d and e denote diagonal and off-diagonal elements of B and vi denotes an element of the vector defining H(i).

Matrix Q can be produced with ReflectTridiagonalToQ method.

© 2000–2025, MetaQuotes Ltd.

ReflectTridiagonalToQ

Generates orthogonal matrix Q which is defined as the product of n-1 elementary reflectors of order n, as returned by ReduceSymmetricToTridiagonal:

in upper case Q = H(n-1) . . . H(2) H(1),

in lower case Q = H(1) H(2) . . . H(n-1).

LAPACK functions ORGTR, UNGTR.

As input is used transformed matrix reflect_q with the same sizes n-by-n as in original matrix A.

Computing for type matrix<double>

```
bool  matrix::ReflectTridiagonalToQ(
   vector&          tau_q,         // scalar factors of the elementary reflectors Q
   matrix&          Q              // matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorf&         tau_q,         // scalar factors of the elementary reflectors Q
   matrixf&         Q              // matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorc&         tau_q,         // scalar factors of the elementary reflectors Q
   matrixc&         Q              // matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReflectTridiagonalToQ(
   vectorcf&        tau_q,         // scalar factors of the elementary reflectors Q
   matrixcf&        Q              // matrix Q
   );
```

Parameters

tau_q

[in] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

Q

[out] Orthogonal matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

ReduceToHessenberg

Reduces a real or complex general n-by-n matrix A to upper Hessenberg form B by an orthogonal similarity transformation: Q**T * A * Q = H. LAPACK function GEHRD.

Computing for type matrix<double>

```
bool  matrix::ReduceToHessenberg(
   matrix&         H,             // upper Hessenberg matrix
   matrix&         reflect_q,     // q-reflectors
   vector&         tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<float>

```
bool  matrixf::ReduceToHessenberg(
   matrixf&        H,             // upper Hessenberg matrix
   matrixf&        reflect_q,     // q-reflectors
   vectorf&        tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<complex>

```
bool  matrixc::ReduceToHessenberg(
   matrixc&        H,             // upper Hessenberg matrix
   matrixc&        reflect_q,     // q-reflectors
   vectorc&        tau_q          // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::ReduceToHessenberg(
   matrixcf&       H,             // upper Hessenberg matrix
   matrixcf&       reflect_q,     // q and p-reflectors
   vectorcf&       tau_q          // scalar factors of the elementary reflectors Q
   );
```

Parameters

H

[out] Upper Hessenberg matrix.

reflect_q

[out] Owerwritten matrix A, the elements below the first subdiagonal, with the array *tau_q*, represent the orthogonal matrix Q as a product of elementary reflectors.

tau_q

[out] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Note

Matrix Q can be produced with ReflectHessenbergToQ method.

© 2000-2025, MetaQuotes Ltd.

ReflectHessenbergToQ

Generates orthogonal matrix Q which is defined as the product of n-1 elementary reflectors of order n, as returned by ReduceToHessenberg:

Q = H(1) H(2) . . . H(n-1).

LAPACK function ORGHR.

As input is used transformed matrix reflect_q with the same sizes n-by-n as in original matrix A.

Computing for type matrix<double>

```
bool  matrix::ReflectHessenbergToQ(
   vector&         tau_q,        // scalar factors of the elementary reflectors Q
   matrix&         Q             // matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrix::ReflectHessenbergToQ(
   vectorf&        tau_q,        // scalar factors of the elementary reflectors Q
   matrixf&        Q             // matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReflectHessenbergToQ(
   vectorc&        tau_q,        // scalar factors of the elementary reflectors Q
   matrixc&        Q             // matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReflectHessenbergToQ(
   vectorcf&       tau_q,        // scalar factors of the elementary reflectors Q
   matrixcf&       Q             // matrix Q
   );
```

Parameters

tau_q

[in] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

Q

[out] Orthogonal matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

Matrix Norm

This section presents functions for computing various matrix norms for different matrix structures—general (rectangular), tridiagonal, upper Hessenberg, symmetric/Hermitian (both full and tridiagonal), and triangular/trapezoidal. Each routine is templated for four data types (double, float, std::complex<double>, std::complex<float>) and overloads the signature

```
bool MatrixNorm*(ENUM_BLAS_NORMX norm, T& norm_value);
```

where norm specifies one of:

- 1-norm (BLASNORMX_O)
- Infinity-norm (BLASNORMX_I)
- Frobenius norm (BLASNORMX_F)
- max-abs element (BLASNORMX_M)

Results are returned via norm_value; the function returns true on success or false on error.

Under the hood, all routines call optimized LAPACK drivers—LANGE, LANGT, LANHS, LANSY/LANHE, LANST/LANHT, and LANTR—to guarantee both high performance and numerical reliability in large-scale linear-algebra computations.

Function	Action
MatrixNorm	Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a general rectangular matrix. LAPACK function LANGE.
MatrixNormGeTrid	Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a general tridiagonal matrix. LAPACK function LANGT.
MatrixNormHessenberg	Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of an upper Hessenberg matrix. LAPACK function LANHS.
MatrixNormSy	Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a real symmetric or complex Hermitian matrix. LAPACK functions LANSY, LANHE.

© 2000–2025, MetaQuotes Ltd.

Function	Action
MatrixNormComplexSy	Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a complexl symmetric (not Hermitian) matrix. LAPACK function LANSY.
MatrixNormSyTrid	Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a real symmetric or complex Hermitian tridiagonal matrix. LAPACK functions LANST, LANHT.
MatrixNormTriangular	Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a trapezoidal m-by-n or triangular matrix. LAPACK function LANTR.

© 2000–2025, MetaQuotes Ltd.

MatrixNorm

Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a general rectangular matrix. LAPACK function LANGE.

Computing for type matrix<double>

```
bool  matrix::MatrixNorm(
   ENUM_BLAS_NORMX norm,           // matrix norm
   double&         norm_value      // matrix norm value
   );
```

Computing for type matrix<float>

```
bool  matrixf::MatrixNorm(
   ENUM_BLAS_NORMX norm,           // matrix norm
   float&          norm_value      // matrix norm value
   );
```

Computing for type matrix<complex>

```
bool  matrixc::MatrixNorm(
   ENUM_BLAS_NORMX norm,           // matrix norm
   double&         norm_value      // matrix norm value
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::MatrixNorm(
   ENUM_BLAS_NORMX norm,           // matrix norm
   float&          norm_value      // matrix norm value
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORMX enumeration, which specifies the value to be returned by the routine.

norm_value

[out] Calculated matrix norm value.

Return Value

Return true if successful, otherwise false in case of an error.

ENUM_BLAS_NORMX

An enumeration defining which norm calculated.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASNORMX_O	'O': One-norm
BLASNORMX_I	'I': Infinity-norm
BLASNORMX_F	'F': Frobenius-norm
BLASNORMX_M	'M': max(abs(A(i,j)))

© 2000-2025, MetaQuotes Ltd.

MatrixNormGeTrid

Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a general tridiagonal matrix. LAPACK function LANGT.

Computing for type matrix<double>

```
bool  matrix::MatrixNormGeTrid(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<float>

```
bool  matrixf::MatrixNormGeTrid(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Computing for type matrix<complex>

```
bool  matrixc::MatrixNormGeTrid(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::MatrixNormGeTrid(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORMX enumeration, which specifies the value to be returned by the routine.

norm_value

[out] Calculated matrix norm value.

Return Value

Return true if successful, otherwise false in case of an error.

ENUM_BLAS_NORMX

An enumeration defining which norm calculated.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASNORMX_O	'O': One-norm
BLASNORMX_I	'I': Infinity-norm
BLASNORMX_F	'F': Frobenius-norm
BLASNORMX_M	'M': max(abs(A(i,j)))

© 2000-2025, MetaQuotes Ltd.

MatrixNormHessenberg

Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of an upper Hessenberg matrix. LAPACK function LANHS.

Computing for type matrix<double>

```
bool  matrix::MatrixNormHessenberg(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<float>

```
bool  matrixf::MatrixNormHessenberg(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Computing for type matrix<complex>

```
bool  matrixc::MatrixNormHessenberg(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::MatrixNormHessenberg(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORMX enumeration, which specifies the value to be returned by the routine.

norm_value

[out] Calculated matrix norm value.

Return Value

Return true if successful, otherwise false in case of an error.

ENUM_BLAS_NORMX

An enumeration defining which norm calculated.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASNORMX_O	'O': One-norm
BLASNORMX_I	'I': Infinity-norm
BLASNORMX_F	'F': Frobenius-norm
BLASNORMX_M	'M': max(abs(A(i,j)))

© 2000-2025, MetaQuotes Ltd.

MatrixNormSy

Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a real symmetric or complex Hermitian matrix. LAPACK functions LANSY, LANHE.

Computing for type matrix<double>

```
bool  matrix::MatrixNormSy(
   ENUM_BLAS_NORMX norm,           // matrix norm
   double&         norm_value      // matrix norm value
   );
```

Computing for type matrix<float>

```
bool  matrixf::MatrixNormSy(
   ENUM_BLAS_NORMX norm,           // matrix norm
   float&          norm_value      // matrix norm value
   );
```

Computing for type matrix<complex>

```
bool  matrixc::MatrixNormSy(
   ENUM_BLAS_NORMX norm,           // matrix norm
   double&         norm_value      // matrix norm value
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::MatrixNormSy(
   ENUM_BLAS_NORMX norm,           // matrix norm
   float&          norm_value      // matrix norm value
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORMX enumeration, which specifies the value to be returned by the routine.

norm_value

[out] Calculated matrix norm value.

Return Value

Return true if successful, otherwise false in case of an error.

ENUM_BLAS_NORMX

An enumeration defining which norm calculated.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASNORMX_O	'O': One-norm
BLASNORMX_I	'I': Infinity-norm
BLASNORMX_F	'F': Frobenius-norm
BLASNORMX_M	'M': max(abs(A(i,j)))

© 2000-2025, MetaQuotes Ltd.

MatrixNormComplexSy

Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a complexl symmetric (not Hermitian) matrix. LAPACK function LANSY.

Computing for type matrix<complex>

```
bool  matrixc::MatrixNormComplexSy(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::MatrixNormComplexSy(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORMX enumeration, which specifies the value to be returned by the routine.

norm_value

[out] Calculated matrix norm value.

Return Value

Return true if successful, otherwise false in case of an error.

ENUM_BLAS_NORMX

An enumeration defining which norm calculated.

ID	Description
BLASNORMX_O	'O': One-norm
BLASNORMX_I	'I': Infinity-norm
BLASNORMX_F	'F': Frobenius-norm
BLASNORMX_M	'M': max(abs(A(i,j)))

© 2000–2025, MetaQuotes Ltd.

MatrixNormSyTrid

Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a real symmetric or complex Hermitian tridiagonal matrix. LAPACK functions LANST, LANHT.

Computing for type matrix<double>

```
bool  matrix::MatrixNormSyTrid(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<float>

```
bool  matrixf::MatrixNormSyTrid(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Computing for type matrix<complex>

```
bool  matrixc::MatrixNormSyTrid(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::MatrixNormSyTrid(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORMX enumeration, which specifies the value to be returned by the routine.

norm_value

[out] Calculated matrix norm value.

Return Value

Return true if successful, otherwise false in case of an error.

ENUM_BLAS_NORMX

An enumeration defining which norm calculated.

© 2000–2025, MetaQuotes Ltd.

ID	Description
BLASNORMX_O	'O': One-norm
BLASNORMX_I	'I': Infinity-norm
BLASNORMX_F	'F': Frobenius-norm
BLASNORMX_M	'M': max(abs(A(i,j)))

© 2000-2025, MetaQuotes Ltd.

MatrixNormTriangular

Returns the value of the 1-norm, infinity-norm, Frobenius norm, or the largest absolute value of any element of a trapezoidal m-by-n or triangular matrix. LAPACK function LANTR.

Computing for type matrix<double>

```
bool  matrix::MatrixNormTriangular(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<float>

```
bool  matrixf::MatrixNormTriangular(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Computing for type matrix<complex>

```
bool  matrixc::MatrixNormTriangular(
   ENUM_BLAS_NORMX norm,          // matrix norm
   double&         norm_value     // matrix norm value
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::MatrixNormTriangular(
   ENUM_BLAS_NORMX norm,          // matrix norm
   float&          norm_value     // matrix norm value
   );
```

Parameters

norm

[in] Value from the ENUM_BLAS_NORMX enumeration, which specifies the value to be returned by the routine.

norm_value

[out] Calculated matrix norm value.

Return Value

Return true if successful, otherwise false in case of an error.

Note

Matrix is triangular if m = n. It can be both upper or lower triangular.

If m < n then matrix must be upper trapezoidal, ie lower m-by-m triangular part must contain zeroes.

If m > n then matrix must be lower trapezoidal, ie upper n-by-n triangular part must contain zeroes.

© 2000–2025, MetaQuotes Ltd.

ENUM_BLAS_NORMX

An enumeration defining which norm calculated.

ID	Description
BLASNORMX_O	'O': One-norm
BLASNORMX_I	'I': Infinity-norm
BLASNORMX_F	'F': Frobenius-norm
BLASNORMX_M	'M': max(abs(A(i,j)))

© 2000–2025, MetaQuotes Ltd.

Matrix Balance

This section presents functions that improve the numerical stability and accuracy of eigenvalue computations by transforming general matrices into balanced forms. Balancing a matrix involves permuting and scaling its rows and columns to reduce its norm and isolate eigenvalues, which can significantly enhance the performance of subsequent algorithms.

These functions are particularly important in preparing matrices for efficient and accurate eigenvalue and Schur decomposition computations. The use of LAPACK functions ensures robust and high-performance linear algebra operations.

Function	Action
MatrixBalance	Balances a general real or complex square matrix A. This involves, first, permuting A by a similarity transformation to isolate eigenvalues in the first 1 to ILO-1 and last IHI+1 to N elements on the diagonal; and second, applying a diagonal similarity transformation to rows and columns ILO to IHI to make the rows and columns as close in norm as possible. Both steps are optional. Balancing may reduce the 1-norm of the matrix, and improve the accuracy of the computed eigenvalues and/or eigenvectors. LAPACK function GEBAL.
EigenVectorsBackward	Forms the right or left eigenvectors of a real or complex general matrix by backward transformation on the computed eigenvectors of the balanced matrix output by MatrixBalance. LAPACK function GEBAK.
ReduceToHessenbergBalanced	Reduces a real or complex general n-by-n balanced matrix A to upper Hessenberg form B by an orthogonal similarity transformation: Q**T * A * Q = H. LAPACK function GEHRD.
ReflectHessenbergBalancedToQ	Generates orthogonal matrix Q which is defined as the product of ihi-ilo elementary reflectors of order n, as returned by ReduceToHessenbergBalanced: Q = H(ilo) H(ilo+1) . . . H(ihi-1). LAPACK function ORGHR.
EigenHessenbergBalancedSchurQ	Computes the eigenvalues of a Hessenberg matrix H and the matrices T and Z from the Schur decomposition H = Z T Z**T, where T is an upper quasi-triangular

© 2000–2025, MetaQuotes Ltd.

Function	Action
	matrix (the Schur form), and Z is the orthogonal matrix of Schur vectors. Optionally Z may be postmultiplied into an input orthogonal matrix Q so that this routine can give the Schur factorization of a matrix A which has been reduced to the Hessenberg form H by the orthogonal matrix Q: A = Q*H*Q**T = (QZ)*T*(QZ)**T. LAPACK function HSEQR.

© 2000–2025, MetaQuotes Ltd.

MatrixBalance

Balances a general real or complex square matrix A. This involves, first, permuting A by a similarity transformation to isolate eigenvalues in the first 1 to ILO-1 and last IHI+1 to N elements on the diagonal; and second, applying a diagonal similarity transformation to rows and columns ILO to IHI to make the rows and columns as close in norm as possible. Both steps are optional. Balancing may reduce the 1-norm of the matrix, and improve the accuracy of the computed eigenvalues and/or eigenvectors. LAPACK function GEBAL.

Computing for type matrix<double>

```
bool  matrix::MatrixBalance(
   ENUM_EIG_BALANCE      job,                    // input matrix balancing method
   matrix&               AB,                     // balanced matrix
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vector&               scale                   // details of permutations and scali
   );
```

Computing for type matrix<float>

```
bool  matrixf::MatrixBalance(
   ENUM_EIG_BALANCE      job,                    // input matrix balancing method
   matrixf&              AB,                     // balanced matrix
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vectorf&              scale                   // details of permutations and scali
   );
```

Computing for type matrix<complex>

```
bool  matrixc::MatrixBalance(
   ENUM_EIG_BALANCE      job,                    // input matrix balancing method
   matrixc&              AB,                     // balanced matrix
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vector&               scale                   // details of permutations and scali
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::MatrixBalance(
   ENUM_EIG_BALANCE      job,                    // input matrix balancing method
   matrixcf&             AB,                     // balanced matrix
   long&                 ilo,                    // subscript of balanced matrix
   long&                 ihi,                    // superscript of balanced matrix
   vectorf&              scale                   // details of permutations and scali
   );
```

Parameters

job

© 2000-2025, MetaQuotes Ltd.

[in] Value from the ENUM_EIG_BALANCE enumeration which determines the need and method for balancing the input matrix.

AB

[out] Balanced matrix.

ilo

[out] Subscript of the balanced matrix.

ihi

[out] Superscript of the balanced matrix.

scale

[out] Vector of details of permutations and scaling when balancing the input matrix.

Details of the permutations and scaling factors applied when balancing A.

If P(j) is the index of the row and column interchanged with row and column j, and D(j) is the scaling factor applied to row and column j, then

scale(j) = P(j), for j = 1,...,ilo-1

= D(j), for j = ilo,...,ihi

= P(j) for j = ihi+1,..., n.

The order in which the interchanges are made is n to ihi+1, then 1 to ilo-1.

Return Value

Return true if successful, otherwise false in case of an error.

ENUM_EIG_BALANCE

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGBALANCE_N	Do not diagonally scale or permute
EIGBALANCE_P	Perform permutations to make the matrix more nearly upper triangular. Do not diagonally scale
EIGBALANCE_S	Diagonally scale the matrix. Do not permute
EIGBALANCE_B	Both diagonally scale and permute

© 2000–2025, MetaQuotes Ltd.

EigenVectorsBackward

Forms the right or left eigenvectors of a real or complex general matrix by backward transformation on the computed eigenvectors of the balanced matrix output by MatrixBalance. LAPACK function GEBAK.

Computing for type matrix<double>

```
bool  matrix::EigenVectorsBackward(
   ENUM_EIG_BALANCE      job,                    // matrix balancing method
   ENUM_EIG_SIDE         side,                   // right or left eigenvectors
   long                  ilo,                    // subscript of balanced matrix
   long                  ihi,                    // superscript of balanced matrix
   vector&               scale,                  // details of permutations and scali
   matrix&               V                       // eigenvectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenVectorsBackward(
   ENUM_EIG_BALANCE      job,                    // input matrix balancing method
   ENUM_EIG_SIDE         side,                   // right or left eigenvectors
   long                  ilo,                    // subscript of balanced matrix
   long                  ihi,                    // superscript of balanced matrix
   vectorf&              scale,                  // details of permutations and scali
   matrixf&              V                       // eigenvectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenVectorsBackward(
   ENUM_EIG_BALANCE      job,                    // input matrix balancing method
   ENUM_EIG_SIDE         side,                   // right or left eigenvectors
   long                  ilo,                    // subscript of balanced matrix
   long                  ihi,                    // superscript of balanced matrix
   vector&               scale,                  // details of permutations and scali
   matrixc&              V                       // eigenvectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenVectorsBackward(
   ENUM_EIG_BALANCE      job,                    // input matrix balancing method
   ENUM_EIG_SIDE         side,                   // right or left eigenvectors
   long                  ilo,                    // subscript of balanced matrix
   long                  ihi,                    // superscript of balanced matrix
   vectorf&              scale,                  // details of permutations and scali
   matrixcf&             V                       // eigenvectors
   );
```

Parameters

© 2000–2025, MetaQuotes Ltd.

job

[in] Value from the ENUM_EIG_BALANCE enumeration which determines the need and method for balancing the input matrix.

side

[in] Value from the ENUM_EIG_SIDE enumeration which determines the need and method for balancing the input matrix.

ilo

[in] Subscript of the balanced matrix. As returned by MatrixBalance.

ihi

[in] Superscript of the balanced matrix. As returned by MatrixBalance.

scale

[in] Vector of details of permutations and scaling when balancing the input matrix. As returned by MatrixBalance.

V

[out] Right or left eigenvectors backtransformed from the input eigenvectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

When computing eigenvalues and eigenvectors, matrix balancing (scaling of rows and columns) is often used to improve numerical stability. This balancing is performed by the GEBAL function, which transforms the matrix into a "nicer" form.

After balancing and computing the eigenvectors of the transformed matrix, the result does not correspond to the original matrix. In this case, GEBAK is called to transform the eigenvectors back into the space of the original (unbalanced) matrix.

This method is applied to the matrix of right or left eigenvectors of balanced matrix. They can be produced by some eigen solver.

ENUM_EIG_BALANCE

An enumeration defining the need to compute eigenvectors.

ID	Description
EIGBALANCE_N	Do not diagonally scale or permute
EIGBALANCE_P	Perform permutations to make the matrix more nearly upper triangular. Do not diagonally scale
EIGBALANCE_S	Diagonally scale the matrix. Do not permute
EIGBALANCE_B	Both diagonally scale and permute

© 2000–2025, MetaQuotes Ltd.

ENUM_EIG_SIDE

An enumeration defining the need to compute right or left eigenvectors.

ID	Description
EIGSIDE_R	'R': right eigenvectors
EIGSIDE_L	'L': left eigenvectors

© 2000–2025, MetaQuotes Ltd.

ReduceToHessenbergBalanced

Reduces a real or complex general n-by-n balanced matrix A to upper Hessenberg form B by an orthogonal similarity transformation: Q**T * A * Q = H. LAPACK function GEHRD.

Computing for type matrix<double>

```
bool  matrix::ReduceToHessenbergBalanced(
   long            ilo,          // subscript of balanced matrix
   long            ihi,          // superscript of balanced matrix
   matrix&         H,            // upper Hessenberg matrix
   matrix&         reflect_q,    // q-reflectors
   vector&         tau_q         // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<float>

```
bool  matrixf::ReduceToHessenbergBalanced(
   long            ilo,          // subscript of balanced matrix
   long            ihi,          // superscript of balanced matrix
   matrixf&        H,            // upper Hessenberg matrix
   matrixf&        reflect_q,    // q-reflectors
   vectorf&        tau_q         // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<complex>

```
bool  matrixc::ReduceToHessenbergBalanced(
   long            ilo,          // subscript of balanced matrix
   long            ihi,          // superscript of balanced matrix
   matrixc&        H,            // upper Hessenberg matrix
   matrixc&        reflect_q,    // q-reflectors
   vectorc&        tau_q         // scalar factors of the elementary reflectors Q
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::ReduceToHessenbergBalanced(
   long            ilo,          // subscript of balanced matrix
   long            ihi,          // superscript of balanced matrix
   matrixcf&       H,            // upper Hessenberg matrix
   matrixcf&       reflect_q,    // q and p-reflectors
   vectorcf&       tau_q         // scalar factors of the elementary reflectors Q
   );
```

Parameters

ilo

[in] Subscript of the balanced matrix. As returned by MatrixBalance.

ihi

© 2000–2025, MetaQuotes Ltd.

[in] Superscript of the balanced matrix. As returned by MatrixBalance.

H

[out] Upper Hessenberg matrix.

reflect_q

[out] Owerwritten balanced matrix A, the elements below the first subdiagonal, with the array *tau_q*, represent the orthogonal matrix Q as a product of elementary reflectors.

tau_q

[out] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

Note

It is assumed that A is already upper triangular in rows and columns 1:ilo-1 and ihi+1:N. ilo and ihi are normally set by a previous call to MatrixBalance; otherwise they should be set to 1 and N respectively.

The matrix Q is represented as a product of (ihi-ilo) elementary reflectors

Q = H(ilo) H(ilo+1) . . . H(ihi-1).

Each H(i) has the form

H(i) = I - tau * v * v**T

where tau is a real scalar, and v is a real vector with v(1:i) = 0, v(i+1) = 1 and v(ihi+1:n) = 0; v(i+2:ihi) is stored on exit in reflect_q(i+2:ihi,i), and tau in tau_q(i)

The contents of reflect_q are illustrated by the following example, with n = 7, ilo = 2 and ihi = 6:

```
input matrix on entry,          reflect_q on exit,

( a   a   a   a   a   a   a )   ( a   a   h   h   h   h   a )
(     a   a   a   a   a   a )   (     a   h   h   h   h   a )
(     a   a   a   a   a   a )   (     h   h   h   h   h   h )
(     a   a   a   a   a   a )   (     v2  h   h   h   h   h )
(     a   a   a   a   a   a )   (     v2  v3  h   h   h   h )
(     a   a   a   a   a   a )   (     v2  v3  v4  h   h   h )
(                     a   )   (                     a   )
```

where a denotes an element of the original matrix A, h denotes a modified element of the upper Hessenberg matrix H, and vi denotes an element of the vector defining H(i).

Matrix Q can be produced with ReflectHessenbergBalancedToQ method applied to the reflect_q matrix.

© 2000–2025, MetaQuotes Ltd.

ReflectHessenbergBalancedToQ

Generates orthogonal matrix Q which is defined as the product of ihi-ilo elementary reflectors of order n, as returned by ReduceToHessenbergBalanced:

Q = H(ilo) H(ilo+1) . . . H(ihi-1).

LAPACK function ORGHR.

As input is used transformed matrix reflect_q with the same sizes n-by-n as in original matrix A.

Computing for type matrix<double>

```
bool  matrix::ReflectHessenbergBalancedToQ(
   long            ilo,          // subscript of balanced matrix
   long            ihi,          // superscript of balanced matrix
   vector&         tau_q,        // scalar factors of the elementary reflectors Q
   matrix&         Q             // matrix Q
   );
```

Computing for type matrix<float>

```
bool  matrix::ReflectHessenbergBalancedToQ(
   long            ilo,          // subscript of balanced matrix
   long            ihi,          // superscript of balanced matrix
   vectorf&        tau_q,        // scalar factors of the elementary reflectors Q
   matrixf&        Q             // matrix Q
   );
```

Computing for type matrix<complex>

```
bool  matrix::ReflectHessenbergBalancedToQ(
   long            ilo,          // subscript of balanced matrix
   long            ihi,          // superscript of balanced matrix
   vectorc&        tau_q,        // scalar factors of the elementary reflectors Q
   matrixc&        Q             // matrix Q
   );
```

Computing for type matrix<complexf>

```
bool  matrix::ReflectHessenbergBalancedToQ(
   long            ilo,          // subscript of balanced matrix
   long            ihi,          // superscript of balanced matrix
   vectorcf&       tau_q,        // scalar factors of the elementary reflectors Q
   matrixcf&       Q             // matrix Q
   );
```

Parameters

ilo

[in] Subscript of the balanced matrix. As returned by MatrixBalance.

ihi

© 2000–2025, MetaQuotes Ltd.

[in] Superscript of the balanced matrix. As returned by MatrixBalance.

tau_q

[in] Vector of the scalar factors of the elementary reflectors which represent the orthogonal matrix Q.

Q

[out] Orthogonal matrix Q.

Return Value

Return true if successful, otherwise false in case of an error.

© 2000–2025, MetaQuotes Ltd.

EigenHessenbergBalancedSchurQ

Computes the eigenvalues of a Hessenberg matrix H and the matrices T and Z from the Schur decomposition H = Z T Z**T, where T is an upper quasi-triangular matrix (the Schur form), and Z is the orthogonal matrix of Schur vectors. Optionally Z may be postmultiplied into an input orthogonal matrix Q so that this routine can give the Schur factorization of a matrix A which has been reduced to the Hessenberg form H by the orthogonal matrix Q:

A = Q*H*Q**T = (QZ)*T*(QZ)**T.

LAPACK function HSEQR.

Computing for type matrix<double>

```
bool  matrix::EigenHessenbergBalancedSchurQ(
   long                 ilo,                 // subscript of balanced matrix
   long                 ihi,                 // superscript of balanced matrix
   matrix&              Q,                   // orthogonal matrix Q
   vectorc&             eigen_values,        // vector of computed eigenvalues
   matrix&              schur_t,             // matrix T in Schur form
   matrix&              schur_z              // matrix Z of Schur vectors
   );
```

Computing for type matrix<float>

```
bool  matrixf::EigenHessenbergBalancedSchurQ(
   long                 ilo,                 // subscript of balanced matrix
   long                 ihi,                 // superscript of balanced matrix
   matrixf&             Q,                   // orthogonal matrix Q
   vectorcf&            eigen_values,        // vector of computed eigenvalues
   matrixf&             schur_t,             // matrix T in Schur form
   matrixf&             schur_z              // matrix Z of Schur vectors
   );
```

Computing for type matrix<complex>

```
bool  matrixc::EigenHessenbergBalancedSchurQ(
   long                 ilo,                 // subscript of balanced matrix
   long                 ihi,                 // superscript of balanced matrix
   matrixc&             Q,                   // orthogonal matrix Q
   vectorc&             eigen_values,        // vector of computed eigenvalues
   matrixc&             schur_t,             // matrix T in Schur form
   matrixc&             schur_z              // matrix Z of Schur vectors
   );
```

Computing for type matrix<complexf>

```
bool  matrixcf::EigenHessenbergBalancedSchurQ(
   long                 ilo,                 // subscript of balanced matrix
   long                 ihi,                 // superscript of balanced matrix
   matrixcf&            Q,                   // orthogonal matrix Q
```

© 2000–2025, MetaQuotes Ltd.

```
   vectorcf&            eigen_values,          // vector of computed eigenvalues
   matrixcf&            schur_t,               // matrix T in Schur form
   matrixcf&            schur_z                // matrix Z of Schur vectors
   );
```

Parameters

ilo

[in] Subscript of the balanced matrix. As returned by MatrixBalance.

ihi

[in] Superscript of the balanced matrix. As returned by MatrixBalance.

Q

[in] Orthogonal matrix Q produced by method ReflectHessenbergBalancedToQ. Matrix Q can be of zero size, in this case Hessenberg matrix (not original matrix A) will be decomposed. If matrix Q is used, then calculated the original matrix A reduced to Hessenberg form (see ReduceToHessenbergBalanced).

eigen_values

[out] Vector of eigenvalues.

schur_t

[out] Upper triangular Schur matrix (Schur form for the input matrix).

schur_z

[out] Matrix of Schur vectors.

Return Value

Return true if successful, otherwise false in case of an error.

Note

It is assumed that A is already upper triangular in rows and columns 1:ilo-1 and ihi+1:N. ilo and ihi are normally set by a previous call to MatrixBalance, and then passed to ReduceToHessenbergBalanced when the matrix output by MatrixBalance is reduced to Hessenberg form. Otherwise they should be set to 1 and N respectively.

Real (non-complex) matrices can have a complex solution. Therefore, the input vector of eigenvalues must be complex. In case of a complex solution, the error code is set to 4019 (ERR_MATH_OVERFLOW). Otherwise, only the real parts of the complex values of the eigenvalue vector should be used.

© 2000–2025, MetaQuotes Ltd.

변환 함수

한 서식에서 다른 서식으로 데이터를 변환하는 기능의 그룹.

NormalizeDouble() 기능은 가격 표시에 필요한 정확도를 제공하므로 특별히 주목해야 합니다. 트레이딩 작동 시, 한 자릿수라도 정확도가 거래 서버가 요구하는 값을 초과할 경우 비정규화된 가격을 사용할 수 없습니다.

기능	액션
CharToString	심볼 코드를 한 문자 문자열로 변환
DoubleToString	수의 값을 지정된 정확도로 텍스트 라인 변환
EnumToString	모든 유형의 열거값을 문자열로 변환
NormalizeDouble	부동 소수점 숫자의 지정된 정확도에 대한 반올림
StringToDouble	숫자 심볼 표현을 포함하는 문자열을 double 유형의 수로 변환
StringToInteger	숫자의 기호를 포함하는 문자열을 긴 유형의 수로 변환
StringToTime	날짜 또는 시간을 포함하는 문자열을 "yyyy.mm.dd [hh:mi]" 형식으로 날짜 시간 유형으로 변환
TimeToString	1970년 1월 1일 이후 경과된 시간(초)을 포함하는 값을 "yyyy.mm.dd hh:mi" 형식의 문자열로 변환
IntegerToString	사전 설정된 길의 문자열로 변환
ShortToString	심볼 코드(유니코드)를 하나의 심볼 문자열로 변환
ShortArrayToString	배열 부분을 문자열로 복사
StringToShortArray	심볼에 따라 문자열을 ushort 유형의 배열에서 선택된 부분으로 복사
CharArrayToString	심볼 코드 (ansi)를 하나의 심볼 배열로 변환
StringToCharArray	유니코드에서 ANSI로 변환된 문자열을 선택한 uchar 형식의 배열 위치로 기호별로 복사
CharArrayToStruct	uchar 유형 문자열을 POD 구조에 복사
StructToCharArray	POD 구조를 uchar 유형 문자열에 복사
ColorToARGB	색상의 ARGB 표현을 수신하기 위해 색 유형을 uint 유형으로 변환.
ColorToString	"R,G,B" 문자열로 색상 값을 변환
StringToColor	색상 이름을 가진 "R,G,B" 문자열 또는 문자열을 색상 유형 값으로 변환
StringFormat	사전 설정 형식에 따라 숫자를 문자열로 변환

추가 참조

코드페이지 사용

© 2000-2025, MetaQuotes Ltd.

CharToString

심볼 코드를 단일 문자열로 변환합니다.

```
string  CharToString(
   uchar  char_code      // 기호의 숫자 코드
   );
```

매개변수

char_code

[in] ANSI 기호의 코드.

값 반환

ANSI 심볼 문자열.

예:

```
string ExtStrArray[224];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 심볼 코드 32(공백)부터 255(UCHAR_MAX)까지의 루프에서,
//--- 현재 코드 페이지에 따라 문자열로 변환된 심볼 코드로 배열을 채웁니다.
   for(int i=32; i<=UCHAR_MAX; i++)
      ExtStrArray[i-32]=CharToString((uchar)i);

//--- 저널의 헤더와 심볼 테이블을 인쇄합니다.
   Print("Table of symbols:");
   ArrayPrint(ExtStrArray,_Digits," | ");
   /*
   result:
   Table of symbols:
   [  0] " " | "!" | """ | "#" | "$" | "%" | "&" | "'" | "(" | ")" | "*" | "+" | ","
   [ 25] "9" | ":" | ";" | "<" | "=" | ">" | "?" | "@" | "A" | "B" | "C" | "D" | "E"
   [ 50] "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "[" | "\" | "]" | "^"
   [ 75] "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w"
   [100] "„" | "…" | "†" | "‡" | "€" | "‰" | "Љ" | "‹" | "Њ" | "Ќ" | "Ћ" | "Џ" | "ђ"
   [125] "ќ" | "ћ" | "џ" | " " | "ў" | "Ў" | "J" | "¤" | "Ґ" | "¦" | "§" | "Ё" | "©"
   [150] "¶" | "·" | "ё" | "№" | "є" | "»" | "ј" | "S" | "s" | "ї" | "A" | "Б" | "B"
   [175] "П" | "Р" | "С" | "Т" | "У" | "Ф" | "Х" | "Ц" | "Ч" | "Ш" | "Щ" | "Ъ" | "Ы"
   [200] "и" | "й" | "к" | "л" | "м" | "н" | "о" | "п" | "р" | "с" | "т" | "у" | "ф"
   */
  }
```

추가 참조

© 2000-2025, MetaQuotes Ltd.

StringToCharArray, ShortToString, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

CharArrayToString

uchar 유형의 배열 일부를 복사하여 반환된 문자열로 변환.

```
string  CharArrayToString(
   uchar  array[],              // 배열
   int    start=0,              // 배열의 시작 포지션
   int    count=-1              // 심볼의 번호
   uint    codepage=CP_ACP      // 코드 페이지
   );
```

매개변수

array[]

[in] uchar 유형 배열.

start=0

[in] 복사가 시작되는 포지션 기본적으로 0이 사용됨.

count=-1

[in] 복사할 배열 요소의 수. 결과 문자열의 길이를 정의. 기본값은 -1이며, 이는 배열 끝까지 복사하거나 터미널 0까지 복사함을 의미합니다.

codepage=CP_ACP

[in] 코드 페이지의 값입니다. 가장 많이 사용되는 코드 페이지에는 다수의 기본 제공 상수가 있습니다.

값 반환

String.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 심볼 코드를 사용하여 배열을 만듭니다.
   uchar array[]= { 84, 104, 105, 115,  32, 105, 115,  32,  97,  32,
                    116, 101, 115, 116,  32, 111, 102,  32, 116, 104,
                    101,  32,  67, 104,  97, 114,  65, 114, 114,  97,
                    121,  84, 111,  83, 116, 114, 105, 110, 103,  40,
                     41,  32, 102, 117, 110,  99, 116, 105, 111, 110
                   };
//--- 저널에 문자열로 변환된 코드 배열을 출력합니다.
   Print(CharArrayToString(array));

//--- 심볼 코드와 터미널 제로를 사용하여 배열을 만듭니다.
   uchar array_z[]= { 84, 104, 105, 115,  32, 105, 115,  32,  97,  32,
```

© 2000-2025, MetaQuotes Ltd.

```
                     116, 101, 115, 116,   0,  32, 111, 102,  32, 116,
                     104, 101,  32,  67, 104,  97, 114,  65, 114, 114,
                      97, 121,  84, 111,  83, 116, 114, 105, 110, 103,
                      40,  41,  32, 102, 117, 110,  99, 116, 105, 111, 110
                    };
//--- 저널에 터미널 00이 있는 두 번째 코드 배열을 출력합니다.
   PrintFormat("%s ...", CharArrayToString(array_z));

   /*
   result:
   This is a test of the CharArrayToString() function
   This is a test ...
   */
  }
```

추가 참조

StringToCharArray, ShortArrayToString, 코드페이지 사용

© 2000-2025, MetaQuotes Ltd.

CharArrayToStruct

uchar 유형 문자열을 POD 구조에 복사.

```
bool  CharArrayToStruct(
   void&          struct_object,     // 구조
   const uchar&   char_array[],      // 배열
   uint           start_pos=0        // 배열의 시작 위치
   );
```

매개변수

struct_object

[in] 모든 유형의 POD 구조 (단순 데이터 유형만 포함)에 대한 참조.

char_array[]

[in] uchar 유형 배열.

start_pos=0

[in] 배열의 포지션, 데이터 복사는 다음부터 시작됩니다.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

예:

```
#define DATA_TOTAL 4

MqlRates ExtDataRates[DATA_TOTAL];
uchar    ExtCharArray[];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 마지막 4개 바의 데이터를 uchar 배열에 복사합니다.
   ResetLastError();
   for(int i=0; i<DATA_TOTAL; i++)
     {
      if(!CopyRatesToCharArray(i, ExtCharArray))
         return;
     }

//--- 사용 가능한 모든 데이터를 uchar 배열의 데이터 양만큼 루프의 MqlRates 구조 배열에 복사합니
   for(int i=0; i<DATA_TOTAL; i++)
     {
      ResetLastError();
      if(!CharArrayToStruct(ExtDataRates[i], ExtCharArray, sizeof(MqlRates)*i))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Print("CharArrayToStruct() failed. Error code: ", GetLastError());
       return;
      }
    //--- uchar 배열에서 MqlRates 구조로 데이터를 성공적으로 복사한 때,
    //--- 구조에 복사된 데이터를 저널에 출력합니다.
    MqlRatesPrint(ExtDataRates[i]);
   }

/*
result:
MqlRates data:
 Time = 2024.02.21 09:00;
 Open = 1.08116;
 High = 1.08141;
 Low = 1.08062;
 Close = 1.08124;
 Tick volume = 5344;
 Spread = 1;
 Real volume = 0

MqlRates data:
 Time = 2024.02.21 08:00;
 Open = 1.08149;
 High = 1.08153;
 Low = 1.08073;
 Close = 1.08114;
 Tick volume = 3607;
 Spread = 2;
 Real volume = 0

MqlRates data:
 Time = 2024.02.21 07:00;
 Open = 1.08143;
 High = 1.08165;
 Low = 1.08122;
 Close = 1.08150;
 Tick volume = 2083;
 Spread = 0;
 Real volume = 0

MqlRates data:
 Time = 2024.02.21 06:00;
 Open = 1.08178;
 High = 1.08190;
 Low = 1.08132;
 Close = 1.08142;
 Tick volume = 1733;
 Spread = 0;
```

© 2000-2025, MetaQuotes Ltd.

```
    Real volume = 0
   */
  }
//+------------------------------------------------------------------+
//| 바 데이터를 uchar 배열에 복사                                    |
//| MqlRates 구조를 통해 지정된 인덱스로                             |
//+------------------------------------------------------------------+
bool CopyRatesToCharArray(const int index, uchar &data_array[])
  {
   MqlRates rates[1];
   uint     start=sizeof(MqlRates);

   ResetLastError();
   if(CopyRates(Symbol(), PERIOD_CURRENT, index, 1, rates)!=1)
     {
      Print("CopyRates() failed. Error code: ", GetLastError());
      return(false);
     }
   if(!StructToCharArray(rates[0], data_array, start*index))
     {
      Print("StructToCharArray() failed. Error code: ", GetLastError());
      return(false);
     }

   return(true);
  }
//+------------------------------------------------------------------+
//| 저널의 MqlRates 구조의 필드를 출력합니다                         |
//+------------------------------------------------------------------+
void MqlRatesPrint(MqlRates &rates)
  {
//--- 출력 문자열 생성
   string text=StringFormat(" Time = %s;\n"
                            " Open = %.*f;\n"
                            " High = %.*f;\n"
                            " Low = %.*f;\n"
                            " Close = %.*f;\n"
                            " Tick volume = %I64u;\n"
                            " Spread = %d;\n"
                            " Real volume = %I64u",
                            TimeToString(rates.time),
                            _Digits, rates.open,
                            _Digits, rates.high,
                            _Digits, rates.low,
                            _Digits, rates.close,
                            rates.tick_volume,
                            rates.spread,
                            rates.real_volume);
//--- 저널의 헤더와 출력 문자열을 인쇄합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("MqlRates data:\n", text, "\n");
  }
```

추가 참조

StringToCharArray, ShortArrayToString, StructToCharArray, 코드페이지의 사용, FileReadStruct, Unions (union), MathSwap

© 2000-2025, MetaQuotes Ltd.

StructToCharArray

POD 구조를 uchar 유형 배열에 복사.

```
bool  StructToCharArray(
   const void&  struct_object,     // 구조
   uchar&       char_array[],      // 배열
   uint         start_pos=0        // 배열의 시작 포지션
   );
```

매개변수

struct_object

[in] 모든 유형의 POD 구조 참조 (단순 데이터 유형만 포함).

char_array[]

[in] uchar 유형 배열.

start_pos=0

[in] 복사된 데이터가 추가되는 배열 위치.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

주의

복사할 때 공간이 부족하면 (ArrayResize) 동적 배열이 자동으로 확장됨. 배열을 필수 값까지 확장할 수 없으면 함수가 오류를 반환.

예:

```
uchar    ExtCharArray[];
MqlRates ExtRates[1];

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 하나의 바의 데이터를 MqlRates 구조에 복사합니다.
   if(CopyRates(Symbol(), PERIOD_CURRENT, 0, 1, ExtRates)!=1)
     {
      Print("CopyRates() failed. Error code: ", GetLastError());
      return;
     }

//--- 저널의 MqlRates 구조로 수신된 데이터를 인쇄합니다.
   Print("Data in the structure immediately after receiving it:");
   MqlRatesPrint(ExtRates[0]);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 구조를 uchar 유형의 배열에 복사합니다.
   ResetLastError();
   if(!StructToCharArray(ExtRates[0], ExtCharArray))
     {
      Print("StructToCharArray() failed. Error code: ", GetLastError());
      return;
     }

//--- 구조를 비웁니다
   ZeroMemory(ExtRates[0]);
//--- 비운 후 구조체의 데이터를 인쇄합니다.
   Print("\nData in the structure after ZeroMemory():");
   MqlRatesPrint(ExtRates[0]);

//--- 이제 uchar 배열의 데이터를 MqlRates 구조에 복사합니다.
   if(!CharArrayToStruct(ExtRates[0], ExtCharArray))
     {
      Print("CharArrayToStruct() failed. Error code: ", GetLastError());
      return;
     }
//--- uchar 배열에서 데이터를 복사한 후 MqlRates 구조의 데이터를 인쇄합니다.
   Print("\nData in the MqlRates structure after copying it from a uchar array:");
   MqlRatesPrint(ExtRates[0]);

   /*
   result:

   Data in the structure immediately after receiving it:
   MqlRates data:
    Time = 2024.02.21 07:00;
    Open = 1.08143;
    High = 1.08158;
    Low = 1.08122;
    Close = 1.08137;
    Tick volume = 1341;
    Spread = 4;
    Real volume = 0

   Data in the structure after ZeroMemory():
   MqlRates data:
    Time = 0;
    Open = 0.00000;
    High = 0.00000;
    Low = 0.00000;
    Close = 0.00000;
    Tick volume = 0;
    Spread = 0;
    Real volume = 0
```

© 2000-2025, MetaQuotes Ltd.

```
   Data in the MqlRates structure after copying it from a uchar array:
   MqlRates data:
    Time = 2024.02.21 07:00;
    Open = 1.08143;
    High = 1.08158;
    Low = 1.08122;
    Close = 1.08137;
    Tick volume = 1341;
    Spread = 4;
    Real volume = 0
   */
  }
//+------------------------------------------------------------------+
//| 저널의 MqlRates 구조의 필드를 출력합니다.                          |
//+------------------------------------------------------------------+
void MqlRatesPrint(MqlRates &rates)
  {
//--- 출력 문자열 생성
   string text=StringFormat(" Time = %s;\n"
                            " Open = %.*f;\n"
                            " High = %.*f;\n"
                            " Low = %.*f;\n"
                            " Close = %.*f;\n"
                            " Tick volume = %I64u;\n"
                            " Spread = %d;\n"
                            " Real volume = %I64u",
                            TimeToString(rates.time),
                            _Digits, rates.open,
                            _Digits, rates.high,
                            _Digits, rates.low,
                            _Digits, rates.close,
                            rates.tick_volume,
                            rates.spread,
                            rates.real_volume);
//--- 저널의 헤더와 출력 문자열을 인쇄합니다.
   Print("MqlRates data:\n", text);
  }
```

추가 참조

StringToCharArray, ShortArrayToString, CharArrayToStruct, Use of a Codepage, FileWriteStruct, Unions (union), MathSwap

© 2000-2025, MetaQuotes Ltd.

ColorToARGB

색상 유형을 uint 유형으로 변환하여 ARGB 표현되는 기능. ARGB 색상 서식은 그래픽 리소스, 텍스트 디스플레이 및 CCanvas 표준 라이브러리 클래스를 생성하는데 사용됩니다.

```
uint  ColorToARGB(
   color  clr,          // 색상 서식의 변환된 색상
   uchar  alpha=255     // 알파 채널 색상 투명도 관리
   );
```

매개변수

clr

[in] 색상 유형 변수의 색상 값.

알파

[in] ARGB 서식의 색상을 수신하는 데 사용되는 알파 채널의 값입니다. 값은 0(전경 픽셀의 색상은 기본 픽셀의 표시를 변경하지 않음)부터 255(기본 픽셀의 색상은 전면 픽셀의 색상으로 완전히 대체됨)까지 설정할 수 있습니다. 백분율 텀의 색상 투명도는 (1-알파/255)*100%로 계산됩니다. 즉 알파 채널의 값이 작을수록 더 투명한 색이 됩니다.

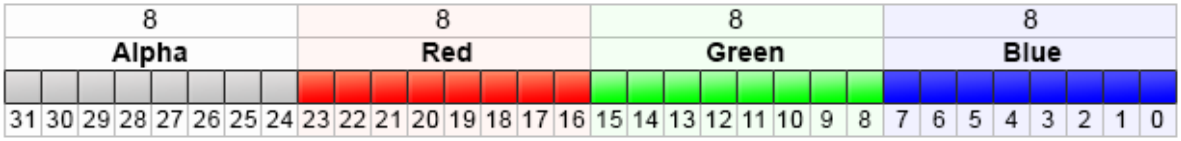
값 반환

Alfa, Red, Green, Blue (알파 채널, red, green, blue) 값이 4 uint 유형 바이트로 직렬로 설정된 경우 ARGB 형식으로 색상을 표시합니다.

주의

RGB는 컴퓨터 그래픽에서 픽셀 색 설명에 일반적으로 사용되는 기본 형식입니다. 기본 색상의 이름은 Red, Green 및 Blue 구성 요소를 설정하는 데 사용됩니다. 각 구성 요소는 색상 포화도를 0 ~255(16진법 형식의 0x00 ~ 0XFF) 범위에서 지정하는 1바이트로 설명됩니다. 흰색은 모든 색상을 포함하므로 0xFFFFFF라고 하며, 즉, 세 구성 요소 각각이 최대값 0xFF로 표시됩니다.

그러나 일부 태스크는 이미지가 어느 정도 투명도를 가진 색상으로 가려지는 경우 이미지의 모양을 설명하는 색상 투명도를 지정해야 합니다. 이러한 경우에 알파 채널 개념을 도입합니다. RGB 형식의 추가 구성 요소로 구현됩니다. ARGB 포맷 구조는 다음과 같습니다.



ARGB 값은 일반적으로 각각 Alpha, Red, Green 및 Blue 채널의 값을 나타내는 각 숫자 쌍과 함께 16진법 형식을 사용하여 표현됩니다. 예를 들어, 80FFFF00 색상은 50.2% 불투명 노란색을 나타냅니다. 0x80은 0xFF 값의 50.2%이므로 처음에는 50.2%의 알파 값을 설정합니다. 그런 다음 첫 번째 FF 쌍은 빨간색 구성 요성의 가장 높은 값을 정의하고, 다음 FF 쌍은 이전과 동일하지만 녹색 구성 요소, 마지막 00 쌍은 파란색 구성 요소의 가장 낮은 값(파란색 없음)을 나타냅니다. 녹색과 빨간색의 조합은 노란색을 낳습니다. 알파 채널을 사용하지 않으면 입력을 6 RRGGBB 자리까지 줄일 수 있습니다. 그렇기 때문에 알파 채널 값이 uint 정수 유형의 상위 비트에 저장됩니다.

컨텍스트에 따라 16진법 자리수는 '0x' 또는 '#' 접두사, 예를 들어, 80FFFF00, 0x80FFFF00 또는 #80FFFF00로 쓸 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 투명도 설정
   uchar alpha=0x55;  // 0x55는 55/255=21.6 % 투명도를 의미합니다
   //--- clrBlue 색상 ARGB로 변환
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB with alpha=0x55 (transparency 21.6%%)",ColorToAR
   //--- clrGreen 색상 ARGB로 변환
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB with alpha=0x55 (transparency 21.6%%)",ColorToA
   //--- clrRed 색상 ARGB로 변환
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB with alpha=0x55 (transparency 21.6%%)",ColorToARG
  }
```

추가 참조

Resources, ResourceCreate(), TextOut(), color type, char, short, int and long types

© 2000-2025, MetaQuotes Ltd.

ColorToString

색상 값을 "R,G,B" 형식 문자열로 변환.

```
string  ColorToString(
   color  color_value,     // 색상 값
   bool   color_name       // 색상명을 표시 또는 숨김
   );
```

매개변수

color_value

[in] 색상 유형 변수의 색상 값.

color_name

[in] 색상 상수가 사전 정의된 색상 상수 중 하나와 동일한 경우 색상 이름을 반환.

값 반환

색상을 "R,G,B", 여기서 R, G 및 B 는 문자열로 변환된 0부터 255까지의 십진법 상수입니다. color_name=true 매개변수를 설정한 경우 색상 값을 색상 이름으로 변환하려고 시도합니다.

예를 들어:

```
   string clr=ColorToString(C'0,255,0'); // 녹색
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // 갯상 상수 가져오기
   Print(clr);
```

추가 참조

StringToColor, ColorToARGB

© 2000-2025, MetaQuotes Ltd.

DoubleToString

숫자 값을 텍스트 문자열로 변환.

```
string  DoubleToString(
   double  value,      // 숫자
   int     digits=8    // 소수점 아래 자릿수
   );
```

매개변수

값

[in] 플로팅 포인트가 있는 값

자릿수

[in] 정확도 서식. *자릿수* 값이 0과 16 사이의 범위에 있는 경우, 지정된 자릿수의 숫자 문자열 표시가 포인트 뒤에 표시됩니다. *자릿수* 값이 -1에서 -16 사이의 범위에 있는 경우, 소수점 뒤에 지정된 자릿수로 구성된 과학적 형식의 숫자를 문자열로 나타냅니다. 다른 모든 경우 문자열 값은 소수점 뒤에 8자리입니다.

값 반환

지정된 정확도의 숫자를 나타내는 기호를 포함하는 문자열.

예를 들어:

```
   Print("DoubleToString(120.0 + M_PI) : ",DoubleToString(120.0+M_PI));
   Print("DoubleToString(120.0 + M_PI,16) : ",DoubleToString(120.0+M_PI,16));
   Print("DoubleToString(120.0 + M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));
   Print("DoubleToString(120.0 + M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));
   Print("DoubleToString(120.0 + M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

추가 참조

NormalizeDouble, StringToDouble

© 2000-2025, MetaQuotes Ltd.

EnumToString

모든 유형의 열거값을 텍스트 형식으로 변환.

```
string  EnumToString(
   any_enum  value      // 열거값의 모든 유형
   );
```

매개변수

값

[in] 열거값의 모든 유형.

값 반환

열거형 텍스트를 나타내는 string. 오류 메시지를 받으려면 GetLastError() 함수를 호출.

주의

함수는 _LastError 변수에서 다음 오류 값을 설정할 수 있습니다:

- ERR_INTERNAL_ERROR – 실행 환경의 오류
- ERR_NOT_ENOUGH_MEMORY – 메모리가 부족하여 작업을 완료할 수 없음
- ERR_INVALID_PARAMETER – 열거형 값의 이름을 허용할 수 없음

예를 들어:

```
enum interval  // 명명된 상수의 열거
  {
   month=1,     // 1개월 간격
   two_months,  // 2개월
   quarter,     // 3개월 - 1분기
   halfyear=6,  // 반년
   year=12,     // 1년 - 12개월
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 시간 간격을 1개월로 설정
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- 시간 간격을 1분기(3개월)로 설정
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- 시간 간격을 1년(12개월)로 설정
   period=year;
```

© 2000-2025, MetaQuotes Ltd.

```
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- 주문 유형이 표시되는 방식을 확인
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- 잘못된 값이 표시되는 방식을 확인
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// 결과:
// 개월 =1
// 분기 =3
// 년=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

추가 참조

Enumerations, 변수 입력

© 2000-2025, MetaQuotes Ltd.

IntegerToString

이 함수는 정수 유형의 값을 지정된 길이의 문자열로 변환하고 가져온 문자열을 반환합니다.

```
string  IntegerToString(
   long    number,              // 숫자
   int     str_len=0,           // 결과 문자열 길이
   ushort  fill_symbol=' '      // 필터
   );
```

매개변수

숫자

[in] 전환 번호.

str_len=0

[in] 문자열 길이. 결과 문자열 길이가 지정된 문자열 길이보다 크면 문자열이 잘리지 않습니다. 크기가 작을 경우 필터 심볼이 왼쪽에 추가됩니다.

fill_symbol=' '

[in] 필터 심볼. 기본적으로 공백입니다.

값 반환

String.

예:

```
#define DATA_TOTAL 1001

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- DATA_TOTAL로 루프에서 인덱스 배수가 100인 행을 표시합니다.
//--- 문자열은 인덱스 값을 4자리 형식으로 표시합니다.
//--- 문자열 길이가 4자 미만인 경우 루프 인덱스 값
//--- 문자열 앞에는 0이 붙습니다.
   for(int i=0; i<DATA_TOTAL; i++)
     {
      if(i%100==0)
         Print("Converted index value: ",IntegerToString(i,4,'0'));
     }
   /*
   result:
   Converted index value: 0000
   Converted index value: 0100
   Converted index value: 0200
   Converted index value: 0300
```

© 2000-2025, MetaQuotes Ltd.

```
   Converted index value: 0400
   Converted index value: 0500
   Converted index value: 0600
   Converted index value: 0700
   Converted index value: 0800
   Converted index value: 0900
   Converted index value: 1000
   */
  }
```

추가 참조

StringToInteger

© 2000-2025, MetaQuotes Ltd.

ShortToString

심볼 코드 (유니코드)를 하나의 심볼 문자열로 변환하고 결과 문자열을 반환합니다.

```
string  ShortToString(
   ushort  symbol_code      // 심볼
   );
```

매개변수

symbol_code

[in] 심볼 코드. 심볼 코드 대신 심볼이 포함된 리터럴 문자열 또는 유니코드 테이블의 기호에 해당하는 2바이트 16진수 코드가 포함된 리터럴 문자열을 사용할 수 있습니다.

값 반환

String.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 유니코드 번호 U+23E9를 가진 문자부터 시작하여 루프에서 18자를 인쇄합니다.
   for(int i=0; i<18; i++)
     {
      ushort code=0x23E9+(ushort)i;
      PrintFormat("Unicode number U+%hX: %s",code,ShortToString(code));
     }
   /*
   result:
   Unicode number U+23E9: ⏩
   Unicode number U+23EA: ⏪
   Unicode number U+23EB: ⏫
   Unicode number U+23EC: ⏬
   Unicode number U+23ED: ⏭
   Unicode number U+23EE: ⏮
   Unicode number U+23EF: ⏯
   Unicode number U+23F0: ⏰
   Unicode number U+23F1: ⏱
   Unicode number U+23F2: ⏲
   Unicode number U+23F3: ⏳
   Unicode number U+23F4: □
   Unicode number U+23F5: □
   Unicode number U+23F6: □
   Unicode number U+23F7: □
   Unicode number U+23F8: ⏸
```

© 2000-2025, MetaQuotes Ltd.

```
   Unicode number U+23F9: ▣
   Unicode number U+23FA: ⦿
   */
  }
```

추가 참조

StringToCharArray, CharToString, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

ShortArrayToString

배열의 일부를 반환된 문자열로 복사.

```
string  ShortArrayToString(
   ushort  array[],      // 배열
   int     start=0,      // 배열의 시작 포지션
   int     count=-1      // 심볼의 번호
   );
```

매개변수

array[]

[in] ushort 유형의 배열(wchar_t type의 아날로그).

start=0

[in] 복사가 시작되는 포지션, 기본값 -0.

count=-1

[in] 복사할 배열 요소의 수. 결과 문자열의 길이를 정의. 기본값은 -1이며, 이는 배열 끝까지 복사하거나 터미널 0까지 복사함을 의미합니다.

값 반환

String.

예:

```
#define ROW_SIZE 16

ushort ExtShortArray[]={0x2190,0x2191,0x2192,0x2193,0x2194,0x2195,0x2196,0x2197,0x2198
                        0x21A0,0x21A1,0x21A2,0x21A3,0x21A4,0x21A5,0x21A6,0x21A7,0x21A8
                        0x21B0,0x21B1,0x21B2,0x21B3,0x21B4,0x21B5,0x21B6,0x21B7,0x21B8
                        0x21C0,0x21C1,0x21C2,0x21C3,0x21C4,0x21C5,0x21C6,0x21C7,0x21C8
                        0x21D0,0x21D1,0x21D2,0x21D3,0x21D4,0x21D5,0x21D6,0x21D7,0x21D8
                        0x21E0,0x21E1,0x21E2,0x21E3,0x21E4,0x21E5,0x21E6,0x21E7,0x21E8
                        0x21F0,0x21F1,0x21F2,0x21F3,0x21F4,0x21F5,0x21F6,0x21F7,0x21F8

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 7x16 심볼 테이블의 문자열 수를 계산합니다.
   int total=(int)ExtShortArray.Size()/ROW_SIZE;
//--- 테이블 행별 루프에서 유니코드 문자로 각 행에 16개 조각씩 7개의 화살표 세트를 저널에 한 행씩
   for(int i=0; i<total; i++)
     {
      int row=i*ROW_SIZE;
      PrintFormat("Arrow set %u: %s",i+1,ShortArrayToString(ExtShortArray,row,ROW_SIZE
```

© 2000-2025, MetaQuotes Ltd.

```
   }
  /*
  result:
  Arrow set 1: ←↑→↓↔↕□□□□□□□□□□
  Arrow set 2: □□□□□□□□↕□□□□□□□
  Arrow set 3: □□□□□□□□□□□□□□□□
  Arrow set 4: □□□□□□□□□□□□□□□□
  Arrow set 5: □□⇒□⇔□□□□□□□□□□
  Arrow set 6: □□□□□□□□□□□□□□□□
  Arrow set 7: □□□□□□□□□□□□□□□□
  */
 }
```

추가 참조

StringToShortArray, CharArrayToString, Use of a Codepage

© 2000-2025, MetaQuotes Ltd.

TimeToString

1970년 1월 1일 이후 경과된 시간(초)을 포함하는 값을 "yyyy.mm.dd hh:mi" 형식의 문자열로 변환.

```
string  TimeToString(
   datetime  value,                          // 숫자
   int       mode=TIME_DATE|TIME_MINUTES     // 서식 출력
   );
```

매개변수

값

[in] 1970년 1월 1일 00:00 부터의 시간(초).

mode=TIME_DATE|TIME_MINUTES

[in] 추가 데이터 입력 모드. 하나 또는 결합된 플래그일 수 있습니다:
TIME_DATE는 "yyyy.mm.dd",
TIME_MINUTES는 "hh:mi",
TIME_SECONDS는 "hh:mi:ss"로 결과를 가져옴.

값 반환

String.

예:

```
datetime ExtBarTimeOpen;

//+------------------------------------------------------------------+
//| 맞춤형 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 타이머를 1초로 세팅
   EventSetTimer(1);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 맞춤형 지표 초기화 해제 함수                                     |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   Comment("");
  }
//+------------------------------------------------------------------+
//| 맞춤형 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
```

© 2000-2025, MetaQuotes Ltd.

```
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 현재 바의 오픈 시간을 가져옵니다.
   ExtBarTimeOpen=time[rates_total-1];
//--- 다음번 호출을 위해 prev_calculated의 값을 반환합니다.
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 타이머 함수                                                      |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- 이전 바의 오픈 시간을 가져옵니다.
   static datetime bar_open_time=ExtBarTimeOpen;
//--- 바가 열린 후 경과한 초의 수를 계산합니다.
   static int seconds=int(TimeCurrent()-ExtBarTimeOpen);
//--- 이전 개장 시간이 현재 개장 시간과 같지 않으면 이는 새로운 바입니다.
//--- 새로운 오픈 시간을 이전 시간으로 쓰고 초를 0으로 설정합니다.
   if(bar_open_time!=ExtBarTimeOpen)
     {
      bar_open_time=ExtBarTimeOpen;
      seconds=0;
     }
//--- 바가 열린 후 경과한 초의 수를 늘리고 조정합니다.
   seconds++;
   if(seconds>PeriodSeconds(PERIOD_CURRENT))
      seconds=0;
//---yyyy.mm.dd hh:mi으로 된 바 오픈 시간(
   string bar_time_open=TimeToString(ExtBarTimeOpen);
//--- yyyy.mm.dd hh:mi:ss으로 된 현재 시간
   string time_current=TimeToString(TimeCurrent(),TIME_DATE|TIME_MINUTES|TIME_SECONDS)
//--- 새로운 바가 열릴 때까지 남은 초의 수
   int    sec_left=PeriodSeconds(PERIOD_CURRENT)-seconds;
//--- hh:mm:ss으로 된 현재 바가 닫힐 때까지 남은 시간
   string time_left=TimeToString(sec_left,TIME_MINUTES|TIME_SECONDS);
//--- 출력 문자열 생성
   string txt=StringFormat("Opening time of the current bar: %s\n"+
                           "Time Current: %s\n"+
                           "Seconds have passed since the bar opened: %d\n"+
                           "Approximately seconds left before bar closes: %d\n"+
                           "Time remaining until bar closes: %s",bar_time_open,time_cu
//--- 바 오픈 시간과 현재 시간을 표시합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 현재 바가 열린 이후 지나간 초와 닫힐 때까지 남은 초
//--- 코멘트에서 현재 바가 닫힐 때까지 남은 시간
   Comment(txt);
   /*
   result on M1:
   Opening time of the current bar: 2024.02.22 18:06
   Time Current: 2024.02.22 18:06:24
   Seconds have passed since the bar opened: 25
   Approximately seconds left before bar closes: 35
   Time remaining until bar closes: 00:00:35

   result on M5:
   Opening time of the current bar: 2024.02.22 18:05
   Time Current: 2024.02.22 18:07:28
   Seconds have passed since the bar opened: 149
   Approximately seconds left before bar closes: 151
   Time remaining until bar closes: 00:02:31

   result on H1:
   Opening time of the current bar: 2024.02.22 18:00
   Time Current: 2024.02.22 18:08:13
   Seconds have passed since the bar opened: 494
   Approximately seconds left before bar closes: 3106
   Time remaining until bar closes: 00:51:46

   result on D1:
   Opening time of the current bar: 2024.02.22 00:00
   Time Current: 2024.02.22 18:11:01
   Seconds have passed since the bar opened: 65462
   Approximately seconds left before bar closes: 20938
   Time remaining until bar closes: 05:48:58
   */
  }
```

추가 참조

StringToTime, TimeToStruct

© 2000-2025, MetaQuotes Ltd.

NormalizeDouble

플로팅 포인트 숫자를 지정된 정확도로 반올림 합니다.

```
double  NormalizeDouble(
   double   value,      // 정규화된 수
   int      digits      // 소수점 아래의 자릿수
   );
```

매개변수

값

[in] 플로팅 포인트가 있는 값.

자릿수

[in] 정확도 형식, 점 뒤의 자릿수(0-8).

값 반환

사전 설정 정확도와 함께 double 유형의 값.

주의

StopLoss, TakeProfit의 계산된 값과 보류 중인 주문에 대한 공개 가격 값은 Digits()로 얻을 수 있는 정확한 값으로 정규화 해야합니다.

Print() 기능을 사용하여 저널에 출력할 때 정규화된 숫자에 예상보다 많은 소수 자릿수가 포함될 수 있습니다. 예를 들어:

```
   double a=76.671;              // 소수점 3자리 정규화 숫자
   Print("Print(76.671)=",a);    // 있는 그대로 출력
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // 사전 설정 정확도록 출력
```

터미널에 다음이 있습니다:

```
   DoubleToString(a,8)=76.67100000

   Print(76.671)=76.67100000000001
```

예를 들어:

```
   double pi=M_PI;
   Print("pi = ",DoubleToString(pi,16));

   double pi_3=NormalizeDouble(M_PI,3);
   Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
   ;
   double pi_8=NormalizeDouble(M_PI,8);
   Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

   double pi_0=NormalizeDouble(M_PI,0);
   Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
```

© 2000-2025, MetaQuotes Ltd.

```
/*
  결과:
   pi= 3.1415926535897931
   NormalizeDouble(pi,3)= 3.1419999999999999
   NormalizeDouble(pi,8)= 3.1415926499999998
   NormalizeDouble(pi,0)= 3.0000000000000000
*/
```

추가 참조

DoubleToString, Real types (double, float), Typecasting

© 2000-2025, MetaQuotes Ltd.

StringToCharArray

심볼 별로 유니코드에서 ANSI로 변환된 문자열을 uchar 유형의 선택된 배열 위치로 복사. 복사된 요소 수를 반환합니다.

```
int  StringToCharArray(
   string  text_string,          // 소스 문자열
   uchar&  array[],              // 배열
   int     start=0,              // 배열의 시작 포지션
   int     count=-1              // 심볼의 번호
   uint    codepage=CP_ACP       // 코드 페이지
   );
```

매개변수

text_string

[in] 복사할 문자열.

array[]

[out] uchar 유형의 배열.

start=0

[in] 복사가 시작되는 포지션 기본 -0.

count=-1

[in] 복사할 배열 요소의 수. 결과 문자열의 길이를 정의. 기본값은 -1이며, 이는 배열 끝까지 복사하거나 터미널 0까지 복사함을 의미합니다. 터미널 0도 수신인 배열에 복사되며, 이 경우 동적 배열의 크기를 문자열 크기로 늘릴 수 있습니다. 동적 배열의 크기가 문자열 길이를 초과하면, 배열 크기가 줄어들지 않습니다.

codepage=CP_ACP

[in] 코드 페이지의 값입니다. 가장 많이 사용되는 코드페이지의 경우 적절한 상수를 제공합니다.

값 반환

복사된 요소의 수.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 변환할 문자열
   string text = "This is a test of the StringToCharArray() function";

//--- 설정된 코드 페이지에 따라 입력 문자열을 uchar 배열로 변환합니다.
   uchar char_array[];
   int   copied=StringToCharArray(text, char_array);
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("String length: %u\nNumber of characters copied (with terminal 0): %d\r
               StringLen(text), copied, text);
//--- 결과 배열을 저널에 출력합니다.
   ArrayPrint(char_array, 0, " | ");
   /*
   result:
   String length: 50
   Number of characters copied (with terminal 0): 51
   Array of characters for the string 'This is a test of the StringToCharArray() funct
   [ 0]  84 | 104 | 105 | 115 |  32 | 105 | 115 |  32 |  97 |  32 | 116 | 101 | 115 |
   [17]  32 | 116 | 104 | 101 |  32 |  83 | 116 | 114 | 105 | 110 | 103 |  84 | 111 |
   [34]  65 | 114 | 114 |  97 | 121 |  40 |  41 |  32 | 102 | 117 | 110 |  99 | 116 |
   */
  }
```

추가 참조

CharArrayToString, StringToShortArray, Use of a Codepage

© 2000-2025, MetaQuotes Ltd.

StringToColor

"R,G,B" 문자열 또는 색상 이름을 가진 문자열을 색상 유형 값으로 변환.

```
color  StringToColor(
   string  color_string      // 색상을 나타내는 문자열
   );
```

매개변수

color_string

[in] 미리 정의된 웹 색상 중 하나의 "R,G,B" 유형 또는 이름을 나타내는 문자열.

값 반환

색상 값.

예를 들어:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- 색상을 약간 변경
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

추가 참조

ColorToString, ColorToARGB

© 2000-2025, MetaQuotes Ltd.

StringToDouble

이 함수는 숫자의 심볼 표시가 포함된 문자열을 double 유형의 숫자로 변환.

```
double  StringToDouble(
   string  value      // 문자열
   );
```

매개변수

값

[in] 문자열은 숫자의 심볼을 나타냄.

값 반환

double 유형의 값.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 변환할 문자열
   string str = "12345.54321";
//--- 입력 문자열을 실수인 doubl e형 변수로 변환합니다.
   double converted=StringToDouble(str);
//--- 결과 숫자를 소수점 이하 8자리까지 저널에 정확하게 표시합니다.
   PrintFormat("The string '%s' is converted to the real number %.8f", str, converted)
   /*
   result:
   The string '12345.54321' is converted to the real number 12345.54321000
   */
  }
```

추가 참조

NormalizeDouble, Real types (double, float), Typecasting

© 2000-2025, MetaQuotes Ltd.

StringToInteger

함수는 숫자의 심볼 표시가 포함된 문자열을 long(정수) 유형의 숫자로 변환.

```
long  StringToInteger(
   string  value      // 문자열
   );
```

매개변수

값

[in] 숫자를 포함하는 문자열.

값 반환

long 유형의 값.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 변환할 문자열
   string str = "12345.54321";
//--- 입력 문자열을 숫자로 긴 유형의 변수로 변환합니다.
   long converted=StringToInteger(str);
//--- 얻은 숫자를 분수 부분을 잘라서 저널에 표시합니다.
   PrintFormat("The string '%s' is converted to the integer number %I64d", str, convei
   /*
   result:
   The string '12345.54321' is converted to the integer number 12345
   */
  }
```

추가 참조

IntegerToString, Real types (double, float), Typecasting

© 2000-2025, MetaQuotes Ltd.

StringToShortArray

함수 심볼은 문자열을 ushort 유형의 지정된 위치에 복사. 복사된 요소의 수를 반환.

```
int  StringToShortArray(
   string  text_string,     // 소스 문자열
   ushort& array[],         // 배열
   int     start=0,         // 배열의 시작 포지션
   int     count=-1         // 심볼의 번호
   );
```

매개변수

text_string

[in] 복사할 문자열

array[]

[out] ushort 유형의 배열(wchar_t 유형의 아날로그).

start=0

[in] 복사가 시작되는 포지션. 기본 -0.

count=-1

[in] 복사할 배열 요소의 수. 결과 문자열의 길이를 정의. 기본값은 -1이며, 이는 배열 끝까지 복사하거나 터미널 0까지 복사함을 의미합니다. 터미널 0도 수신인의 배열에 복사되며, 이 경우 동적 배열의 크기를 문자열 크기로 늘릴 수 있습니다. 동적 배열의 크기가 문자열 길이를 초과하면, 배열 크기가 줄어들지 않습니다.

값 반환

복사된 요소의 수.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 변환할 문자가 포함된 문자열
   string text = "Chars: ♡♫☎✈☩☏ ❁✉☆ ⌓♕ ♠®✧ ❄♣    ";
   ushort short_array[];
//--- 입력 문자열에서 ":" 의 위치를 찾아 다음 문자부터 시작하는 부분적인 문자열을 추출합니다.
   int    pos=StringFind(text, ":");
   string str=(pos<0 ? text : StringSubstr(text, pos+1));
//--- 결과 문자열의 왼쪽과 오른쪽에서 공백, 캐리지 이동 및 도표화 문자를 제거합니다.
   StringTrimLeft(str);
   StringTrimRight(str);
//--- 결과 문자열을 ushort 배열에 복사하고 결과 배열을 저널에 인쇄합니다.
   int copied=StringToShortArray(str, short_array);
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("String of characters length: %u\n"
               "Number of characters copied (with terminal 0): %d\n"
               "Array of chars for the string '%s':",
               StringLen(str), copied, str);
   ArrayPrint(short_array, 0, " | ");
   /*
   result:
   String of characters length: 16
   Number of characters copied (with terminal 0): 17
   Array of chars for the string '♡♫☎☄✢☏⌬✉☆☁♕♠®✧❄♣':
   10084 |  9835 |  9742 |  9992 | 10019 |  9743 |  9728 |  9993 |  9734 |  9729 |  98
   */
  }
```

추가 참조

ShortArrayToString, StringToCharArray, Use of a Codepage

© 2000-2025, MetaQuotes Ltd.

StringToTime

"yyyy.mm.dd [hh:mi]" 서식의 시간 및 날짜를 포함하는 문자열을 datetime 유형 번호로 변환.

```
datetime  StringToTime(
   const string  time_string      // 날짜 문자열
   );
```

매개변수

time_string

[in] 지정된 서식 중 하나의 문자열:

- "yyyy.mm.dd [hh:mi]"
- "yyyy.mm.dd [hh:mi:ss]"
- "yyyymmdd [hh:mi:ss]"
- "yyyymmdd [hhmiss]"
- "yyyy/mm/dd [hh:mi:ss]"
- "yyyy-mm-dd [hh:mi:ss]"

값 반환

datetime 유형 값으로 1970.01.01 이후 경과된 시간(초)을 포함합니다.

주의

날짜와 시간 사이의 공백 및 테이블 문자 시퀀스는 ***time_string***이 추가 처리되지 않도록 StringToTime() 호출하기 전에 단일 공백으로 간주됩니다.

예:

```
//--- 입력 매개변수
input group    "The date can be entered in any of the formats:"
input group    "yyyy.mm.dd [hh:mi], yyyy.mm.dd [hh:mi:ss]"
input group    "yyyymmdd [hh:mi:ss], yyyymmdd [hhmiss]"
input group    "yyyy/mm/dd [hh:mi:ss], yyyy-mm-dd [hh:mi:ss]"
input string   InpDateStr;    // Please enter the date here as a string

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력에 문자열로 입력된 시간을 날짜/시간 값으로 변환합니다.
   datetime time=StringToTime(InpDateStr);
//--- 입력한 문자열과 얻은 시간을 저널에 표시합니다.
   PrintFormat("Date entered as a string in the form '%s' is converted to datetime in
               InpDateStr, TimeToString(time, TIME_DATE|TIME_MINUTES|TIME_SECONDS));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 수신된 날짜-시간에 수직선을 생성하고 차트를 이 위치로 이동합니다.
   if(CreateVLine(time))
      ChartNavigateToTime(time);
   /*
   result:
   Date entered as a string in the form '' is converted to datetime in the form '1970.
   Date entered as a string in the form '2024' is converted to datetime in the form '2
   Date entered as a string in the form '202400' is converted to datetime in the form
   Date entered as a string in the form '20240000' is converted to datetime in the for
   Date entered as a string in the form '2024022410' is converted to datetime in the
   Date entered as a string in the form '20240224 10' is converted to datetime in the
   Date entered as a string in the form '20240224 01' is converted to datetime in the
   Date entered as a string in the form '20240224 0030' is converted to datetime in th
   Date entered as a string in the form '20240224 0100' is converted to datetime in th
   */
  }
//+------------------------------------------------------------------+
//| 수직선 객체 만들기                                               |
//+------------------------------------------------------------------+
bool CreateVLine(const datetime line_time)
  {
   ResetLastError();

   string name=MQLInfoString(MQL_PROGRAM_NAME)+"_VLINE";
   if(!ObjectCreate(0, name, OBJ_VLINE, 0, line_time, 0))
     {
      Print("ObjectCreate() failed. Error code: ", GetLastError());
      return(false);
     }
   ObjectSetInteger(0, name, OBJPROP_STYLE, STYLE_DOT);
   ObjectSetInteger(0, name, OBJPROP_SELECTABLE, true);

   return(true);
  }
//+------------------------------------------------------------------+
//| 차트를 지정된 바 오픈 시간으로 이동                              |
//+------------------------------------------------------------------+
bool ChartNavigateToTime(const datetime time)
  {
   ChartSetInteger(0, CHART_AUTOSCROLL, false);
   ResetLastError();

   int bar=iBarShift(_Symbol, PERIOD_CURRENT, time);
   if(bar<0)
     {
      PrintFormat("%s: iBarShift() failed. Error code: %d", __FUNCTION__, GetLastError
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   long first=0;
   if(!ChartGetInteger(0, CHART_FIRST_VISIBLE_BAR, 0, first))
     {
      PrintFormat("%s: ChartGetInteger() failed. Error code: %d", __FUNCTION__, GetLas
      return(false);
     }

   return(ChartNavigate(0, CHART_CURRENT_POS, (int)first-bar));
  }
```

추가 참조

TimeToString, TimeToStruct

© 2000-2025, MetaQuotes Ltd.

StringFormat

함수 서식이 매개변수를 얻어 문자열을 반환.

```
string  StringFormat(
   string  format,      // 설명 서식의 문자열
   ...     ...          // 매개변수
   );
```

매개변수

서식

[in] 서식지정 방식을 포함하는 문자열. 서식 지정 규칙은 PrintFormat 기능과 동일.

...

[in] 쉼표로 구분되는 매개변수.

값 반환

String.

예를 들어:

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- string 변수
   string output_string;
   string temp_string;
   string format_string;
//--- 사양 헤더 준비
   temp_string=StringFormat("Contract specification for %s:\n",_Symbol);
   StringAdd(output_string,temp_string);
//--- int 값 출력
   int digits=(int)SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);
   temp_string=StringFormat("   SYMBOL_DIGITS = %d (number of digits after the decimal
                            digits);
   StringAdd(output_string,temp_string);
//--- 소수점 아래 래 자릿수가 가변적인 double 값 출력
   double point_value=SymbolInfoDouble(_Symbol,SYMBOL_POINT);
   format_string=StringFormat("   SYMBOL_POINT = %%.%df (point value)\n",
                              digits);
   temp_string=StringFormat(format_string,point_value);
   StringAdd(output_string,temp_string);
//--- int 값 출력
   int spread=(int)SymbolInfoInteger(_Symbol,SYMBOL_SPREAD);
   temp_string=StringFormat("   SYMBOL_SPREAD = %d (current spread in points)\n",
                            spread);
   StringAdd(output_string,temp_string);
//--- int 값 출력
   int min_stop=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_STOPS_LEVEL = %d (minimal indention in po
                            min_stop);
   StringAdd(output_string,temp_string);
//--- 부분 부분이 없는 double 값 출력
   double contract_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_CONTRACT_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_CONTRACT_SIZE = %.f (contract size)\n",
                            contract_size);
   StringAdd(output_string,temp_string);
//--- 기본 정확도의 double 값 출력
   double tick_size=SymbolInfoDouble(_Symbol,SYMBOL_TRADE_TICK_SIZE);
   temp_string=StringFormat("   SYMBOL_TRADE_TICK_SIZE = %f (minimal price change)\n",
                            tick_size);
   StringAdd(output_string,temp_string);
//--- 스왑 계산 모드 결정
   int swap_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_SWAP_MODE);
   string str_swap_mode;
   switch(swap_mode)
     {
      case SYMBOL_SWAP_MODE_DISABLED: str_swap_mode="SYMBOL_SWAP_MODE_DISABLED (no swa
      case SYMBOL_SWAP_MODE_POINTS: str_swap_mode="SYMBOL_SWAP_MODE_POINTS (in points)
      case SYMBOL_SWAP_MODE_CURRENCY_SYMBOL: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_MARGIN: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY_
      case SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT: str_swap_mode="SYMBOL_SWAP_MODE_CURRENCY
      case SYMBOL_SWAP_MODE_INTEREST_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST
      case SYMBOL_SWAP_MODE_INTEREST_OPEN: str_swap_mode="SYMBOL_SWAP_MODE_INTEREST_OP
      case SYMBOL_SWAP_MODE_REOPEN_CURRENT: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_CUR
      case SYMBOL_SWAP_MODE_REOPEN_BID: str_swap_mode="SYMBOL_SWAP_MODE_REOPEN_BID (by
     }
//--- string 값 출력
   temp_string=StringFormat("   SYMBOL_SWAP_MODE = %s\n",
                            str_swap_mode);
   StringAdd(output_string,temp_string);
//--- 기본 정확도 double 값 출력
   double swap_long=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_LONG);
```

© 2000-2025, MetaQuotes Ltd.

```
   temp_string=StringFormat("   SYMBOL_SWAP_LONG = %f (long swap value)\n",
                            swap_long);
   StringAdd(output_string,temp_string);
//--- 기본 정확도 double 값 출력
   double swap_short=SymbolInfoDouble(_Symbol,SYMBOL_SWAP_SHORT);
   temp_string=StringFormat("   SYMBOL_SWAP_SHORT = %f (short swap value)\n",
                            swap_short);
   StringAdd(output_string,temp_string);
//--- 트레이딩 모드 결정
   int trade_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_MODE);
   string str_trade_mode;
   switch(trade_mode)
     {
      case SYMBOL_TRADE_MODE_DISABLED: str_trade_mode="SYMBOL_TRADE_MODE_DISABLED (tra
      case SYMBOL_TRADE_MODE_LONGONLY: str_trade_mode="SYMBOL_TRADE_MODE_LONGONLY (onl
      case SYMBOL_TRADE_MODE_SHORTONLY: str_trade_mode="SYMBOL_TRADE_MODE_SHORTONLY (o
      case SYMBOL_TRADE_MODE_CLOSEONLY: str_trade_mode="SYMBOL_TRADE_MODE_CLOSEONLY (o
      case SYMBOL_TRADE_MODE_FULL: str_trade_mode="SYMBOL_TRADE_MODE_FULL (no trade re
     }
//--- string 값 출력
   temp_string=StringFormat("   SYMBOL_TRADE_MODE = %s\n",
                            str_trade_mode);
   StringAdd(output_string,temp_string);
//--- 압축 서식의 double 값 출력
   double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   temp_string=StringFormat("   SYMBOL_VOLUME_MIN = %g (minimal volume for a deal)\n",
   StringAdd(output_string,temp_string);
//--- 압축 서식의 double 값 출력
   double volume_step=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_STEP);
   temp_string=StringFormat("   SYMBOL_VOLUME_STEP = %g (minimal volume change step)\n
   StringAdd(output_string,temp_string);
//--- 압축 서식의 double 값 출력
   double volume_max=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MAX);
   temp_string=StringFormat("   SYMBOL_VOLUME_MAX = %g (maximal volume for a deal)\n",
   StringAdd(output_string,temp_string);
//--- 계약 가격 계산 모드 결정
   int calc_mode=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_CALC_MODE);
   string str_calc_mode;
   switch(calc_mode)
     {
      case SYMBOL_CALC_MODE_FOREX:str_calc_mode="SYMBOL_CALC_MODE_FOREX (Forex)";breal
      case SYMBOL_CALC_MODE_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_FUTURES (futures)"
      case SYMBOL_CALC_MODE_CFD:str_calc_mode="SYMBOL_CALC_MODE_CFD (CFD)";break;
      case SYMBOL_CALC_MODE_CFDINDEX:str_calc_mode="SYMBOL_CALC_MODE_CFDINDEX (CFD for
      case SYMBOL_CALC_MODE_CFDLEVERAGE:str_calc_mode="SYMBOL_CALC_MODE_CFDLEVERAGE (C
      case SYMBOL_CALC_MODE_EXCH_STOCKS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_STOCKS (t
      case SYMBOL_CALC_MODE_EXCH_FUTURES:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FUTURES
      case SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS:str_calc_mode="SYMBOL_CALC_MODE_EXCH_FU
     }
//--- string 값 출력
   temp_string=StringFormat("   SYMBOL_TRADE_CALC_MODE = %s\n",
                            str_calc_mode);
   StringAdd(output_string,temp_string);
//--- 소수점 아래 2자리 숫자가 있는 double 값 출력
   double margin_initial=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_INITIAL);
   temp_string=StringFormat("   SYMBOL_MARGIN_INITIAL = %.2f (initial margin)\n",
                            margin_initial);
   StringAdd(output_string,temp_string);
//--- 소수점 아래 2자리 숫자가 있는 double 값 출력
   double margin_maintenance=SymbolInfoDouble(_Symbol,SYMBOL_MARGIN_MAINTENANCE);
   temp_string=StringFormat("   SYMBOL_MARGIN_MAINTENANCE = %.2f (maintenance margin)\
```

© 2000-2025, MetaQuotes Ltd.

```
                              margin_maintenance);
   StringAdd(output_string,temp_string);
//--- int 값 출력
   int freeze_level=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_FREEZE_LEVEL);
   temp_string=StringFormat("   SYMBOL_TRADE_FREEZE_LEVEL = %d (order freeze level in
                              freeze_level);
   StringAdd(output_string,temp_string);
   Print(output_string);
   Comment(output_string);
/* 실행 결과
   Contract specification for EURUSD:
     SYMBOL_DIGITS = 5 (number of digits after the decimal point)
     SYMBOL_POINT = 0.00001 (point value)
     SYMBOL_SPREAD = 10 (current spread in points)
     SYMBOL_TRADE_STOPS_LEVEL = 18 (minimal indention in points for Stop orders)
     SYMBOL_TRADE_CONTRACT_SIZE = 100000 (contract size)
     SYMBOL_TRADE_TICK_SIZE = 0.000010 (minimal price change)
     SYMBOL_SWAP_MODE = SYMBOL_SWAP_MODE_POINTS (in points)
     SYMBOL_SWAP_LONG = -0.700000 (buy order swap value)
     SYMBOL_SWAP_SHORT = -1.000000 (sell order swap value)
     SYMBOL_TRADE_MODE = SYMBOL_TRADE_MODE_FULL (no trade restrictions)
     SYMBOL_VOLUME_MIN = 0.01 (minimal volume for a deal)
     SYMBOL_VOLUME_STEP = 0.01 (minimal volume change step)
     SYMBOL_VOLUME_MAX = 500 (maximal volume for a deal)
     SYMBOL_TRADE_CALC_MODE = SYMBOL_CALC_MODE_FOREX (Forex)
     SYMBOL_MARGIN_INITIAL = 0.00 (initial margin)
     SYMBOL_MARGIN_MAINTENANCE = 0.00 (maintenance margin)
     SYMBOL_TRADE_FREEZE_LEVEL = 0 (order freeze level in points)
*/
  }
```

추가 참조

PrintFormat, DoubleToString,ColorToString, TimeToString

© 2000-2025, MetaQuotes Ltd.

수학적 함수

수학 및 삼각함수의 집합

수학 함수는 원래는 스칼라 값에 대한 관련 연산을 수행하도록 설계되었습니다. 이 빌드에서는 대부분의 함수를 행렬과 벡터에 적용할 수 있습니다. 다음을 포함합니다. MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctan. 이러한 연산은 행렬 또는 벡터의 요소별 처리를 의미합니다. 예시:

```
//---
  matrix a= {{1, 4}, {9, 16}};
  Print("matrix a=\n",a);
  a=MathSqrt(a);
  Print("MatrSqrt(a)=\n",a);
  /*
   matrix a=
   [[1,4]
    [9,16]]
   MatrSqrt(a)=
   [[1,2]
    [3,4]]
  */
```

MathMod와 MathPow의 경우 두 번째 요소는 적절한 크기의 벡터/행렬 또는 스칼라일 수 있습니다.

기능	액션
MathAbs	지정된 숫자 값의 절대값(계수)를 반환
MathArccos	라디안 단위 x의 아크 코사인을 반환
MathArcsin	라디안 단위로 x의 아크 사인을 반환
MathArctan	라디안 단위의 x의 아크 탄젠트를 반환
MathArctan2	탄젠트가 지정된 두 숫자의 몫인 각도(라디안 단위)를 반환
MathClassify	실수 유형을 반환
MathCeil	위에서 가장 가까운 정수 숫자 값을 반환
MathCos	수의 코사인을 반환
MathExp	숫자의 지수를 반환
MathFloor	아래에서 가장 가까운 정수 숫자 값을 반환
MathLog	자연 로그 반환
MathLog10	상용 로그의 수를 반환
MathMax	두 숫자 값의 최대값을 반환
MathMin	두 숫자 값의 최소값을 반환

© 2000-2025, MetaQuotes Ltd.

기능	액션
MathMod	두 숫자의 나눗셈 후 실제 나머지를 반환
MathPow	베이스를 지정된 파워로 올림
MathRand	0 부터 32767까지 범위 내 유사 랜덤 값을 반환
MathRound	가장 가까운 정수로 값 반올림
MathSin	숫자의 사인을 반환
MathSqrt	제곱근을 반환
MathSrand	일련의 유사 랜덤 정수를 생성하기 위한 시작점을 설정
MathTan	수의 탄젠트를 반환
MathIsValidNumber	실수의 정확성을 확인
MathExpm1	표현식 MathExp(x)-1의 값을 반환
MathLog1p	표현식 MathLog(1+x)의 값을 반환
MathArccosh	쌍곡선 아크코사인을 반환
MathArcsinh	쌍곡선 아크사인을 반환
MathArctanh	쌍곡선 아크탄젠트를 반환
MathCosh	쌍곡선 코사인을 반환
MathSinh	쌍곡선 사인을 반환
MathTanh	쌍곡선 탄젠트를 반환
MathSwap	ushort/uint/ushort 유형 값에서 바이트 주문 변경

© 2000-2025, MetaQuotes Ltd.

MathAbs

함수는 지정된 숫자 값의 절대값(계수)을 반환.

```
double  MathAbs(
   double  value      // 숫자 값
   );
```

매개변수

값

[in] 숫자 값.

값 반환

0보다 크거나 같은 double 형식의 값

주의

MathAbs() 함수 대신 fabs()를 사용할 수 있습니다.

예:

```
//--- 입력 매개변수
input int      InpValue = -10;   // 아무 값이나 여기에 입력

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력에 입력된 숫자의 절대값을 가져옵니다.
   uint abs_value=MathAbs(InpValue);
//--- 값을 저널에 출력
PrintFormat("입력된 값 %d는 MathAbs() 함수를 사용한 후 %u 값을 받았습니다.",InpValue,abs_va
  }
```

© 2000-2025, MetaQuotes Ltd.

MathArccos

이 함수는 The function returns the arccosine of x within the range 0 부터 π까지의 라디안 범위 내에서 x의 아크코사인을 반환.

```
double  MathArccos(
   double  val     // -1<val<1
   );
```

매개변수

val

[in] 계산되어야 하는 아크코사인의 -1과 1 사이의 val 값.

값 반환

라디안 단위의 수의 아크 코사인. 만약 val 이 -1보다 작거나 1보다 큰 경우, 함수는 NaN (미확정 값)을 반환.

주의

MathArccos() 함수 대신 acos()을 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/1000);
   delta[0]=-1;
//--- 델타 스텝을 사용하여 -1에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 아크코사인 값을 계산합니다.
   vector Y=MathArccos(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

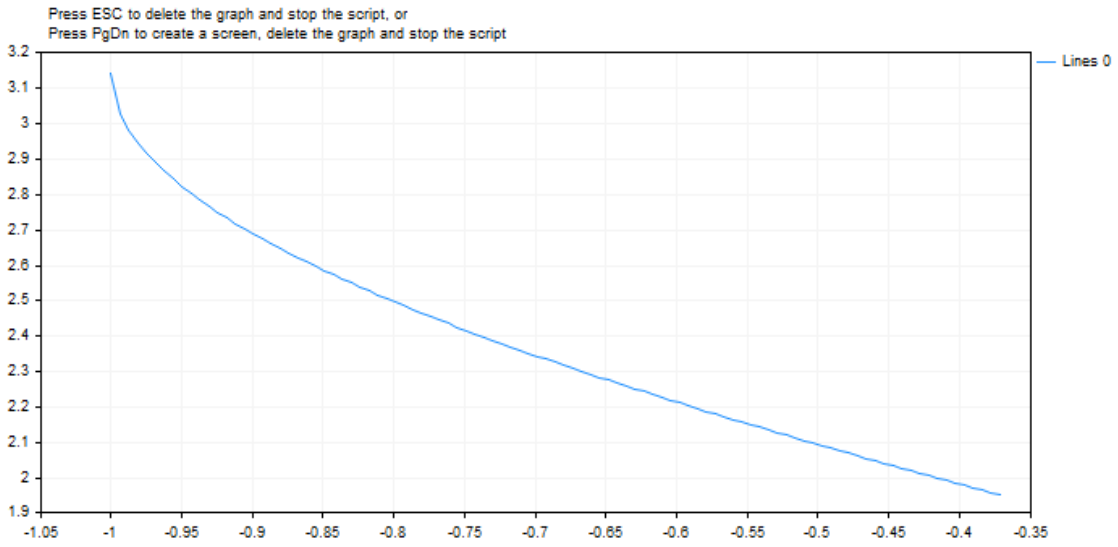
© 2000-2025, MetaQuotes Ltd.

```
//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



추가 참조

실제 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArcsin

이 함수는 -π/2 에서 π/2 라디안까지의 범위 내에서 x의 아크 사인을 반환.

```
double  MathArcsin(
double val      // -1<값<1
   );
```

매개변수

val

[in] 계산되어야 하는 아크 사인 -1과 1 사이의 val 값.

값 반환

-π/2 에서 π/2 라디안까지의 범위 내에서 val 숫자의 아크 사인. val이 -1보다 작거나 1보다 크면 함수는 NaN(미확정 값)을 반환합니다.

주의

MathArcsin() 함수 대신에 asin() 함수를 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/1000);
   delta[0]=-1;
//--- 델타 스텝을 사용하여 -1에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 사인 값을 계산합니다.
   vector Y=MathArcsin(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

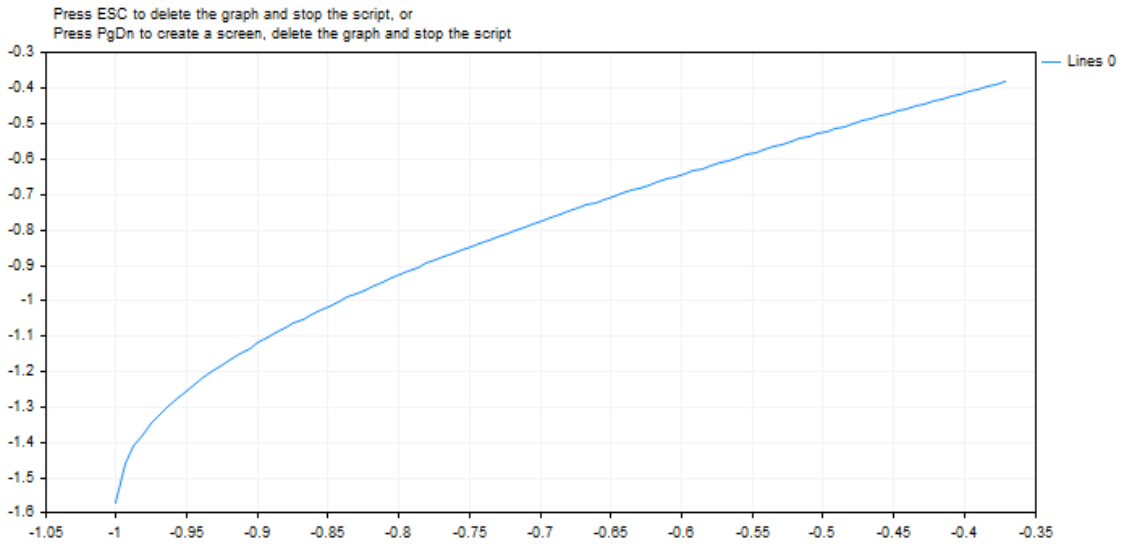
© 2000-2025, MetaQuotes Ltd.

```
//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



추가 참조

실제 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArctan

함수는 x의 아크 탄젠트를 반환. 만약 x 가 0이라면 함수는 0을 반환합니다.

```
double  MathArctan(
   double  value      // 탄젠트
   );
```

매개변수

값

[in] 탄젠트를 나타내는 수

값 반환

MathArctan은 -π/2 에서 π/2 라디안까지의 범위 내 값을 반환.

주의

MathArctan() 함수 대신 **atan**() 함수를 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- 델타 단계를 사용하여 0에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 아크 탄젠트 값을 계산합니다.
   vector Y=MathArctan(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

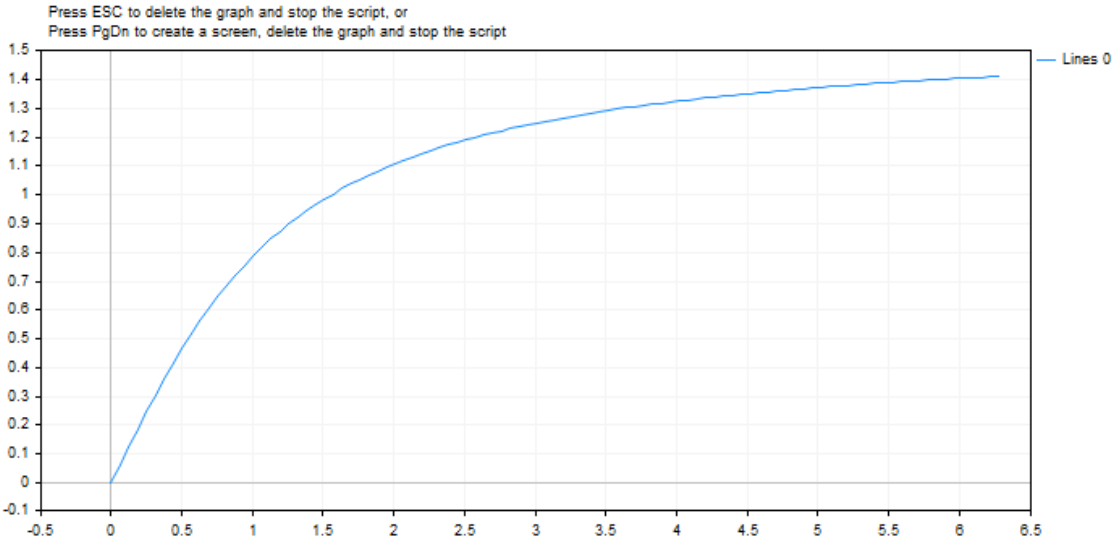
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



© 2000-2025, MetaQuotes Ltd.

MathArctan2

탄젠트가 지정된 두 숫자의 몫인 각도(라디안 단위)를 반환

```
double MathArctan2(
 double y     // 점의 y 좌표
 double x     // 점의 x좌표
  );
```

매개변수

y

[in] Y 좌표 값.

x

[in] X 좌표 값.

값 반환

MathArctan2는 -π 에서 π 라디안까지, MathTan(θ)=y/x 인 각도 θ를 반환

다음을 주의하십시오:

- 1사분면 내 (x, y)의 경우 0< θ < π/2
- 2사분면 내 (x, y)의 경우 π/2< θ ≤ π
- 3사분면 내 (x, y)의 경우 -π < θ < -π/2
- 4사분면 내 (x, y)의 경우 -π/2 < θ < 0

사분면의 경계에 있는 점의 경우 반환 값은 다음과 같습니다:

- y가 0이고 x가 음수가 아닌 경우, θ = 0.
- y가 0이고 x가 음수인 경우, θ = π.
- y가 양수이고 x가 0인 경우, θ = π/2.
- y가 음수이고 x가 0인 경우, θ = -π/2.
- y가 0이고 x도 0인 경우, θ = 0.

주의

MathArctan2() 함수 대신 atan2() 함수를 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

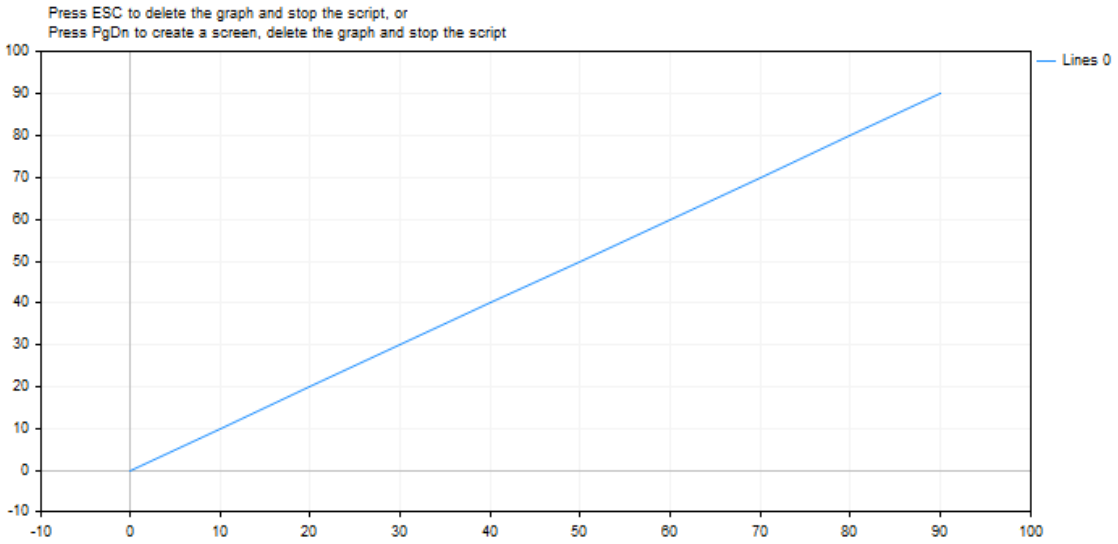
```
  {
   vector delta=vector::Full(10,10);
   delta[0]=0;
//--- 델타 단계를 사용하여 0에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 아크 탄젠트 값을 계산합니다.
   vector Y=delta.CumSum();

   Print("vector delta = \n",delta);
   Print("vector X = \n",X);
   Print("vector Y = \n",Y);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[];;
   double y_array[];;
   X.Swap(x_array);
   Y.Swap(y_array);

   double array[10];
   for(int i=0; i<10; i++)
     {
      array[i]=MathArctan2(y_array[i],x_array[i]);
     }

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
```

© 2000-2025, MetaQuotes Ltd.

```
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
string text1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
string text2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

MathClassify

실수의 유형을 결정하고 ENUM_FP_CLASS 열거값으로 결과를 반환하십시오

```
ENUM_FP_CLASS  MathClassify(
   double  value      // 실수
   );
```

Parameter

value

[in] 확인할 실수

반환값

ENUM_FP_CLASS 열거값

ENUM_FP_CLASS

ID	설명
FP_SUBNORMAL	0에 가까우며 정상수로 표현이 가능한 가장 작은 준정규값 DBL_MIN (2.2250738585072014e-308)
FP_NORMAL	2.2250738585072014e-308 and 1.7976931348623158e+308 범위의 정규값
FP_ZERO	양 또는 음의 0
FP_INFINITE	적절한 유형(양수 또는 음수 무한대)으로 나타낼 수 없는 숫자
FP_NAN	숫자가 아닙니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- NaN 테스트
  double nan=double("nan");
  PrintFormat("Test NaN: %G is %s, MathIsValidNumber(NaN)=%s",
              nan,
              EnumToString(MathClassify(nan)),
              (string)MathIsValidNumber(nan));
//--- 무한대 테스트
  double inf=double("inf");
  PrintFormat("Test Inf: %G is %s, MathIsValidNumber(inf)=%s",
              inf,
              EnumToString(MathClassify(inf)),
              (string)MathIsValidNumber(inf));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 정규값 테스트
  double normal=1.2345e6;
  PrintFormat("Test Normal: %G is %s, MathIsValidNumber(normal)=%s",
              normal,
              EnumToString(MathClassify(normal)),
              (string)MathIsValidNumber(normal));
//--- 비정규값 테스트
  double sub_normal=DBL_MIN/2.0;
  PrintFormat("Test Subnormal: %G is %s, MathIsValidNumber(sub_normal)=%s",
              sub_normal,
              EnumToString(MathClassify(sub_normal)),
              (string)MathIsValidNumber(sub_normal));
//--- 0 값 테스트
  double zero=0.0/(-1);
  PrintFormat("Test Zero: %G is %s, MathIsValidNumber(zero)=%s",
              zero,
              EnumToString(MathClassify(zero)),
              (string)MathIsValidNumber(zero));
 }
 /*
결과:
   Test NaN: NAN is FP_NAN, MathIsValidNumber(NaN)=false
   Test Inf: INF is FP_INFINITE, MathIsValidNumber(inf)=false
   Test Normal: 1.2345E+06 is FP_NORMAL, MathIsValidNumber(normal)=true
   Test Subnormal: 1.11254E-308 is FP_SUBNORMAL, MathIsValidNumber(sub_normal)=true
   Test Zero: -0 is FP_ZERO, MathIsValidNumber(zero)=true
*/
//+------------------------------------------------------------------+
```

참고 항목

Real types (double, float), MathIsValidNumber

© 2000-2025, MetaQuotes Ltd.

MathCeil

함수는 위에서 가장 가까운 정수 값을 반환합니다.

```
double  MathCeil(
   double  val      // 숫자
   );
```

매개 변수

val

[in] 숫자값.

반환값

값을 초과하거나 같은 최소 정수를 나타내는 숫자 값.

참고

MathCeil() 함수 대신에 ceil()을 사용하실 수 있습니다.

예:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 변환을 위한 변수 선언
   double value=0;                     // MathCeil 변환을 위한 실수
   int    ceil_value=0;                // 여기서 결과를 얻습니다.
//--- 실수의 소수점 증분 수만큼 루프에서
   for(int i=0; i<VALUES_TOTAL; i++)
     {
//--- 숫자 값을 늘리고,
//--- 위에서 가장 가까운 정수 값을 가져옵니다.
//--- 그리고 저널에 제어 값을 표시합니다.
      value+=0.1;
      ceil_value=(int)MathCeil(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, ceil value: %d",value,ceil_value);
      /*
     결과:
      value: 0.1, ceil value: 1
      value: 0.2, ceil value: 1
      value: 0.3, ceil value: 1
      value: 0.4, ceil value: 1
      value: 0.5, ceil value: 1
      value: 0.6, ceil value: 1
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, ceil value: 1
      value: 0.8, ceil value: 1
      value: 0.9, ceil value: 1
      value: 1.0, ceil value: 1
      value: 1.1, ceil value: 2
      value: 1.2, ceil value: 2
      value: 1.3, ceil value: 2
      value: 1.4, ceil value: 2
      value: 1.5, ceil value: 2
      value: 1.6, ceil value: 2
      value: 1.7, ceil value: 2
      value: 1.8, ceil value: 2
      value: 1.9, ceil value: 2
      value: 2.0, ceil value: 2
      value: 2.1, ceil value: 3
      value: 2.2, ceil value: 3
      value: 2.3, ceil value: 3
      value: 2.4, ceil value: 3
      value: 2.5, ceil value: 3
      value: 2.6, ceil value: 3
      value: 2.7, ceil value: 3
      value: 2.8, ceil value: 3
      value: 2.9, ceil value: 3
      value: 3.0, ceil value: 3
      value: 3.1, ceil value: 4
      */
    }
}
```

© 2000-2025, MetaQuotes Ltd.

MathCos

함수는 각도의 코사인을 반환합니다.

```
double  MathCos(
   double  value      // 숫자
   );
```

매개 변수

값

[in] 라디안 각도.

반환값

-1 ~1 범위의 double 타입의 값입니다.

참고

MathCos() 대신에 cos()를 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- 델타 단계를 사용하여 0에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 코사인 값을 계산합니다.
   vector Y=MathCos(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

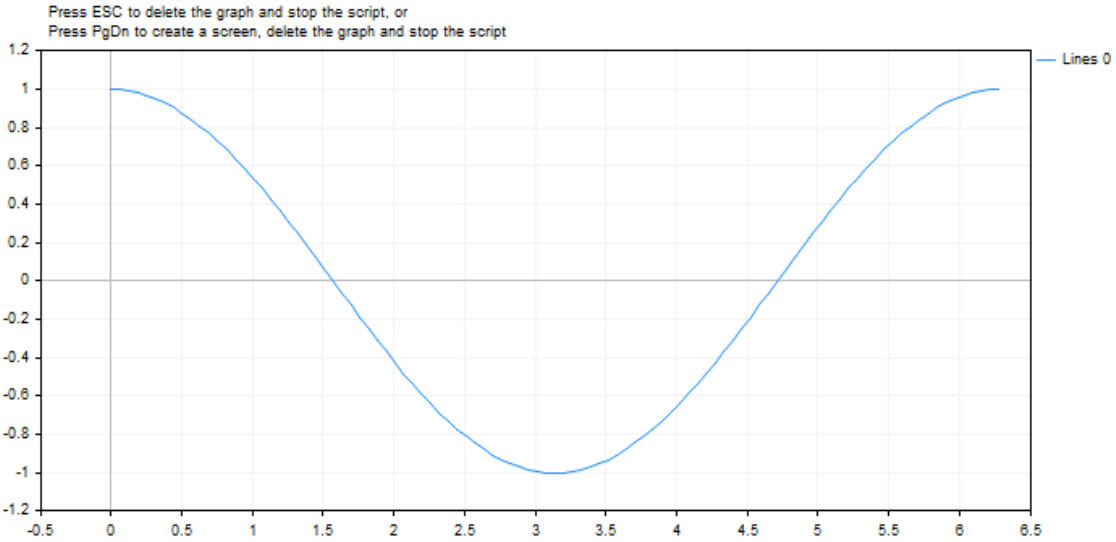
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



© 2000-2025, MetaQuotes Ltd.

MathExp

함수는 e의 상승 값을 d의 거듭제곱으로 반환합니다.

```
double  MathExp(
   double  value      // e의 거듭제곱
   );
```

매개 변수

값

[in] 거듭제곱을 지정하는 숫자.

반환값

double 타입의 수. 오버 플로의 경우 함수는 INF(무한대)를 반환하며, 언더 플로의 경우 MathExp는 0을 반환합니다.

참고

MathExp() 대신에exp()를 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 1 스텝에서 0부터 8까지 9개의 값을 얻습니다.
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- 각 X 벡터 값의 거듭제곱으로 "e"(오일러 수)를 계산합니다.
   X=MathExp(X);
   Print("MathExp(X) = \n",X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double y_array[];
   X.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

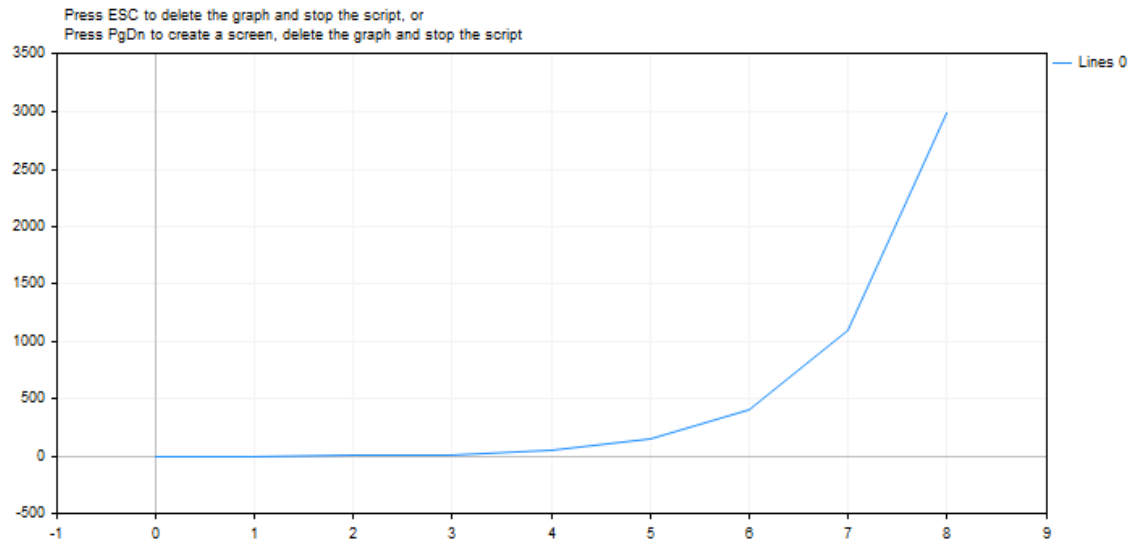
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| '스텝' 증분 단위인 '값'으로 벡터 채우기                                  |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                      |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다                      |
//| 그렇지 않으면 'false'를 반환 합니다                                    |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                        |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                                |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



참고 항목

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathFloor

함수는 그 이하에서 가장 가까운 정수 값을 반환합니다.

```
double  MathFloor(
   double  val     // number
   );
```

매개 변수

val

[in] 숫자값.

반환값

값보다 작거나 같은 가장 큰 정수를 나타내는 값입니다.

참고

MathFloor() 대신에 floor()를 사용하실 수 있습니다.

예:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 변환을 위한 변수 선언
   double value=0;                     // MathFloor 변환을 위한 실수
   int    floor_value=0;               // 여기서 결과를 얻음 here
//--- 실수의 소수점 증분 수만큼 루프에서
   for(int i=0; i<VALUES_TOTAL; i++)
     {
//--- 숫자 값을 늘리고,
//--- 아래에서 가장 가까운 정수 값을 가져옵니다.
//--- 그리고 저널에 제어 값을 표시합니다.
      value+=0.1;
      floor_value=(int)MathFloor(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, floor value: %d",value,floor_value);
      /*
     결과:
      value: 0.1, floor value: 0
      value: 0.2, floor value: 0
      value: 0.3, floor value: 0
      value: 0.4, floor value: 0
      value: 0.5, floor value: 0
      value: 0.6, floor value: 0
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, floor value: 0
      value: 0.8, floor value: 0
      value: 0.9, floor value: 0
      value: 1.0, floor value: 1
      value: 1.1, floor value: 1
      value: 1.2, floor value: 1
      value: 1.3, floor value: 1
      value: 1.4, floor value: 1
      value: 1.5, floor value: 1
      value: 1.6, floor value: 1
      value: 1.7, floor value: 1
      value: 1.8, floor value: 1
      value: 1.9, floor value: 1
      value: 2.0, floor value: 2
      value: 2.1, floor value: 2
      value: 2.2, floor value: 2
      value: 2.3, floor value: 2
      value: 2.4, floor value: 2
      value: 2.5, floor value: 2
      value: 2.6, floor value: 2
      value: 2.7, floor value: 2
      value: 2.8, floor value: 2
      value: 2.9, floor value: 2
      value: 3.0, floor value: 3
      value: 3.1, floor value: 3
      */
     }
```

© 2000-2025, MetaQuotes Ltd.

MathLog

함수는 자연 로그를 반환합니다.

```
double  MathLog(
   double  val      // 로그값
   );
```

매개 변수

val

[in] 찾을 로그 값.

반환값

성공할 경우 발에 대한 자연 로그. 값이 음수이면 함수는 NaN(미결정 값)을 반환합니다. 값이 0이면 함수는 INF(무한대)를 반환합니다.

참고

MathLog() 대신에 log()를 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 1 스텝에서 0부터 8까지 9개의 값을 얻습니다.
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- X 벡터의 각 값에 대한 로그를 계산합니다.
   X=MathLog(X);
   Print("MathLog(X) = \n",X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double y_array[];
   X.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

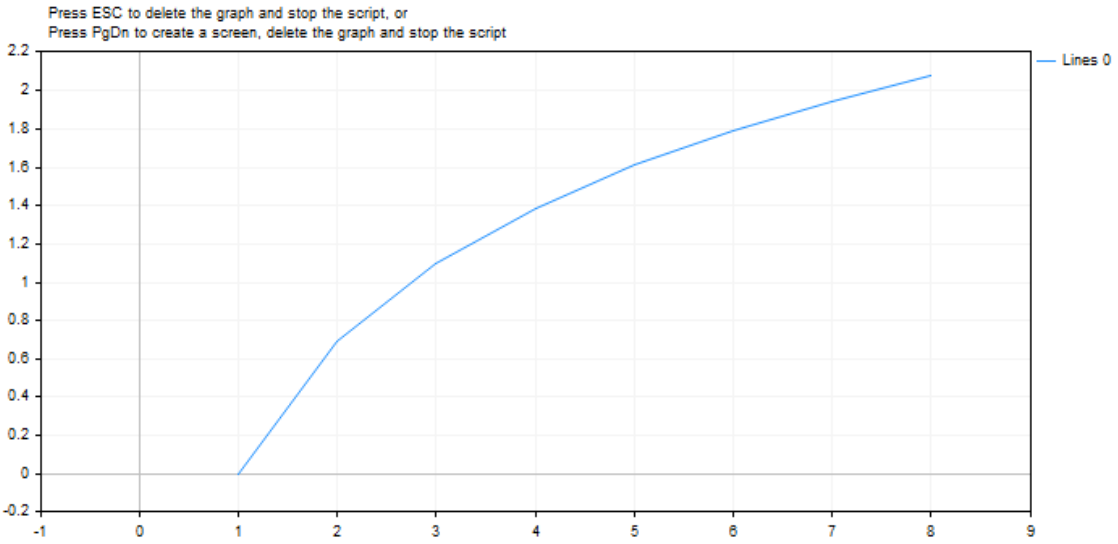
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
   /*
  결과:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog(X) =
   [-inf,0,0.6931471805599453,1.09861228866811,1.386294361119891,1.6094379124341,1.791
   */
  }
//+------------------------------------------------------------------+
//| '스텝' 증분 단위인 '값'으로 벡터 채우기                                  |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                      |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다                      |
//| 그렇지 않으면 'false'를 반환 합니다                                    |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                        |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                                |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



© 2000-2025, MetaQuotes Ltd.

참고 항목

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathLog

숫자 10을 기반으로 한 로그값을 반환합니다.

```
double  MathLog10(
   double  val      // 로그를 씌울 값
   );
```

매개 변수

val

[in] 계산할 상용 로그 값.

반환값

상용로그 결과 값. 값이 음수이면 함수는 NaN(미결정 값)을 반환합니다. 값이 0이면 함수는 INF(무한대)를 반환합니다.

참고

MathLog10() 대신에 log10()을 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 1 스텝에서 0부터 8까지 9개의 값을 얻습니다.
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- X 벡터의 각 값에 대한 십진 로그를 계산합니다.
   X=MathLog10(X);
   Print("MathLog10(X) = \n",X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double y_array[];
   X.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

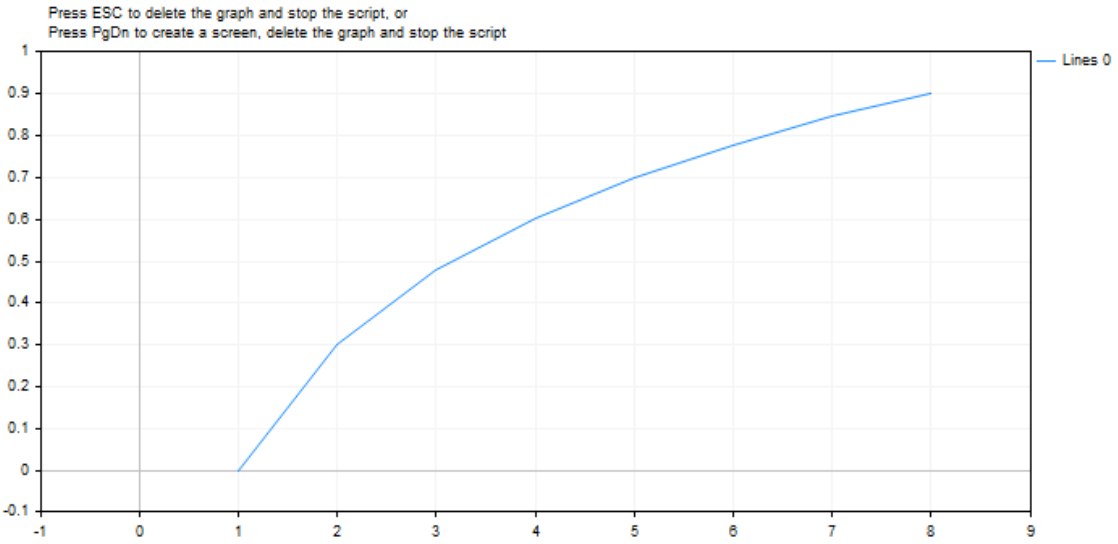
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
   /*
  결과:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog10(X) =
   [-inf,0,0.3010299956639812,0.4771212547196624,0.6020599913279624,0.6989700043360189
   */
  }
//+------------------------------------------------------------------+
//| '스텝' 증분 단위인 '값'으로 벡터 채우기                                  |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                      |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다                      |
//| 그렇지 않으면 'false'를 반환 합니다                                    |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQLI
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                        |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                                |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



© 2000-2025, MetaQuotes Ltd.

참고 항목

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathMax

이 함수는 두 값의 최대값을 반환합니다.

```
double  MathMax(
   double  value1,     // 첫번째 값
   double  value2      // 두번째 값
   );
```

매개 변수

value1

[in] 첫번째 숫자 값.

value2

[in] 두번째 숫자 값.

반환값

두 값 중 더 큰 값.

참고

MathMax() 대신에 fmax()를 사용하실 수 있습니다. 함수 fmax(), fmin(), MathMax(), MathMin()는 정수 형식을 double 형식으로 타이프캐스팅하지 않고 사용할 수 있습니다.

다른 유형의 파라미터가 함수에 전달되면 작은 유형의 파라미터가 큰 유형에 자동으로 캐스트됩니다. 반환 값의 유형은 보다 큰 유형에 해당합니다.

동일한 유형의 데이터를 통과하면 캐스팅이 수행되지 않습니다.

예:

```
//--- 입력 매개변수
input int                  InpPeriod = 10;            // 이동 평균 계산 기간
input ENUM_MA_METHOD       InpMethod = MODE_SMA;      // 이동 평균 계산 방법
input ENUM_APPLIED_PRICE   InpPrice  = PRICE_CLOSE;   // 이동 평균 계산 가격

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 이동 평균 기간을 1보다 작은 값으로 설정하면 기본값(10)이 사용됩니다.
   int period=(InpPeriod<1 ? 10 : InpPeriod);
//--- 이동 평균 지표 핸들 생성
   int handle=iMA(Symbol(),Period(),period,0,InpMethod,InpPrice);
   if(handle==INVALID_HANDLE)
     {
Print("이동 평균 지표 핸들을 생성하지 못했습니다. Error ",GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 현재 Bid 가격을 가져옵니다.
   double bid=0;
   ResetLastError();
   if(!SymbolInfoDouble(Symbol(),SYMBOL_BID,bid))
     {
Print("Bid 가격을 가져오지 못했습니다. Error ",GetLastError());
      return;
     }
//--- 현재 바의 이동 평균 값을 가져옵니다.
   double array[1];
   int    copied=CopyBuffer(handle,0,0,1,array);
   if(copied!=1)
     {
Print("이동 평균 데이터를 가져오지 못했습니다. Error ",GetLastError());
      return;
     }
//--- 둘 중 가장 높은 가격(Bid 가격과 이동 평균 값)을 구하고 결과 데이터를 저널에 표시합니다.
   double max_price=MathMax(bid,array[0]);
   PrintFormat("Bid: %.*f, Moving Average: %.*f, highest price of the two: %.*f",_Digi
   PrintFormat("Bid price %s moving average",(bid>array[0] ? "higher" : bid<array[0] ?
  }
```

© 2000-2025, MetaQuotes Ltd.

MathMin

이 함수는 두 값의 최소값을 반환합니다.

```
double  MathMin(
   double  value1,     // 첫번째 값
   double  value2      // 두번째 값
   );
```

매개 변수

value1

[in] 첫번째 숫자 값.

value2

[in] 두번째 숫자 값.

반환값

두 값 중 가장 작은 값.

참고

MathMin() 대신에 fmin()을 사용할 수 있습니다. 함수 fmax(), fmin(), MathMax(), MathMin() 는 정수 형식을 더블 형식으로 타입캐스팅하지 않고 사용할 수 있습니다.

다른 유형의 파라미터가 함수에 전달되면 작은 유형의 파라미터가 큰 유형에 자동으로 캐스트됩니다. 반환 값의 유형은 보다 큰 유형에 해당합니다.

동일한 유형의 데이터를 통과하면 캐스팅이 수행되지 않습니다.

예:

```
//--- 입력 매개변수
input uint  InpBars  =  100000;  // 원하는 바의 수

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 서버에서 사용 가능한 바의 수를 가져옵니다.
   uint bars_total = Bars(Symbol(),Period());
   if(bars_total==0)
     {
      PrintFormat("Data for timeseries %s %s not yet generated in the terminal. Please
      return;
     }
//--- 두 값(서버에서 사용 가능한 값과 설정에서 요청한 값)에서 최소 바의 수를 가져옵니다.
   int bars = (int)MathMin(bars_total,InpBars);
//--- 서버에서 사용할 수 있는 것보다 더 많은 바가 요청되면 이를 저널에 보고합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
    if(bars_total<InpBars)
       PrintFormat("Number of bars on the server (%u) is less than requested (%u)",bars
//--- 저널에서 작업에 사용할 수 있는 바의 수를 표시합니다.
    Print("Bars available for work: ",bars);
   }
```

© 2000-2025, MetaQuotes Ltd.

MathMod

함수는 두 개의 숫자로 나눈 나머지 부분을 반환합니다.

```
double  MathMod(
   double  value,      // 나눈 몫
   double  value2      // 나눗수
   );
```

매개 변수

값

[in] 나눈 몫.

value2

[in] 나눗수.

반환값

MathMod 함수는 val = i * y + f, 여기서 i는 정수이고 f는 val과 같은 부호를 가지며 f의 절대값이 y의 절대값보다 작도록 식 val/y로부터의 나머지 실수 값 f를 계산합니다.

참고

MathMod() 대신에 fmod()를 사용하실 수 있습니다.

예:

```
#property script_show_inputs

//--- 입력 매개변수
input double   InpDividentValue  =  10;   // Dividend value
input double   InpDivisorValue   =  3;    // Divisor value

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력에 입력된 숫자의 나눗셈의 실수 나머지를 구합니다.
   double res=MathMod(InpDividentValue,InpDivisorValue);
//--- 결과를 저널에 인쇄합니다.
   PrintFormat("Real remainder when dividing %.2f by %.2f = %.2f",InpDividentValue,Inp
  }
```

© 2000-2025, MetaQuotes Ltd.

MathPow

이 함수는 밑을 지수로 올립니다.

```
double  MathPow(
   double  base,          // 밑
   double  exponent       // 지수 값
   );
```

매개 변수

base

[in] 밑.

exponent

[in] 지수 값.

반환값

지정된 거듭제곱으로 올려질 밑의 값.

참고

MathPow() 대신에 pow()를 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#property script_show_inputs

#include <Graphics\Graphic.mqh>

//--- 입력 매개변수
input double   InpExponentValue  =  2;     // exponent value

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 1 스텝에서 0부터 10까지 11개의 값을 얻습니다.
   vector X(11,VectorArange);
   Print("vector X = \n",X);
//--- InpExponentValue 거듭제곱으로 된 X 벡터의 각 값을 계산합니다.
   X=MathPow(X,InpExponentValue);
   Print("MathPow(X,",(string)InpExponentValue,") = \n",X);
```

© 2000-2025, MetaQuotes Ltd.

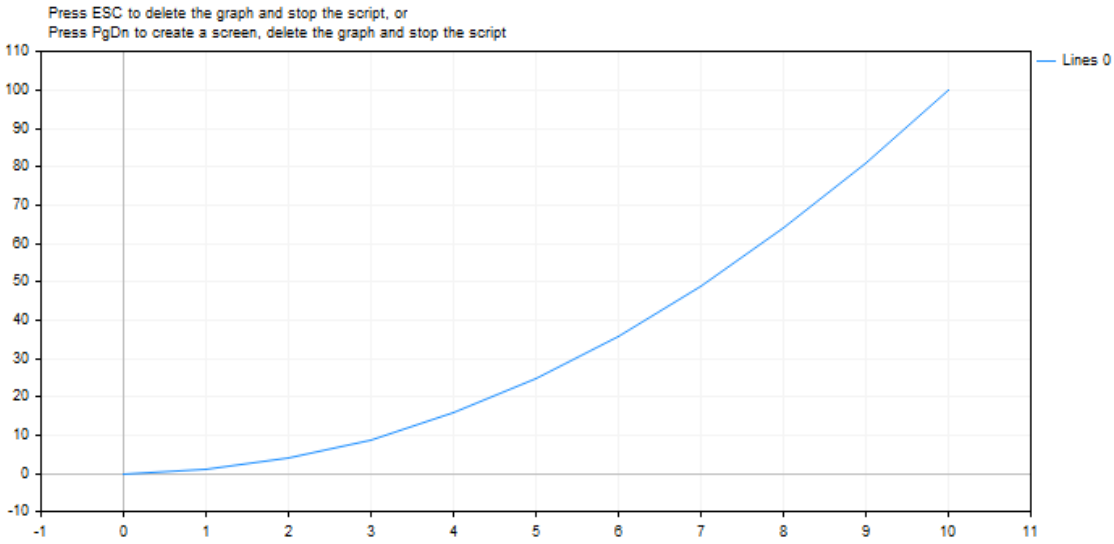
```
//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double y_array[];
   X.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
   /*
  결과:
   vector X =
   [0,1,2,3,4,5,6,7,8,9,10]
   MathPow(X,2.0) =
   [0,1,4,9,16,25,36,49,64,81,100]
   */
  }
//+------------------------------------------------------------------+
//| '스텝' 증분 단위인 '값'으로 벡터 채우기                                  |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                      |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다                      |
//| 그렇지 않으면 'false'를 반환 합니다                                    |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:

© 2000-2025, MetaQuotes Ltd.



© 2000-2025, MetaQuotes Ltd.

MathRand

0 - 32767 범위의 무작위 랜덤 정수를 반환합니다.

```
int  MathRand();
```

반환값

0 - 32767 범위의 정수 값.

참고

함수를 처음 호출하기 전에, MathSrand를 호출하여 무작위 난수 생성기를 초기 상태로 설정해야 합니다.

참고

MathRand() 대신에 rand()를 사용하실 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 시작할 때마다 일련의 의사 난수 정수를 생성하도록 새로운 초기 상태를 설정합니다.
   MathSrand(GetTickCount());
//--- 루프에서 생성된 의사 난수 정수 10개를 저널에 표시합니다.
   for(int i=0; i<10; i++)
     {
      int rand_value=MathRand();
      PrintFormat("Pseudorandom integer №%d: %u",i+1,rand_value);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

MathRound

이 함수는 가장 가까운 특정 정수 값으로 반올림된 값을 반환합니다.

```
double  MathRound(
   double  value      // 반올림될 값
   );
```

매개 변수

값

[in] 반올림 전 숫자 값.

반환값

값이 가장 가까운 정수로 반올림됩니다.

참고

MathRound() 대신에 round()를 사용하실 수 있습니다.

예:

```
#define VALUES_TOTAL 31

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 변환을 위한 변수 선언
   double value=0;                     // real number for MathRound conversion
   int    round_value=0;               // 여기서 결과를 얻습니다
//--- 실수의 소수점 증분 수만큼 루프에서
   for(int i=0; i<VALUES_TOTAL; i++)
     {
//--- 실수 값을 증가시키고,
//--- 가장 가까운 정수로 반올림된 숫자 값을 얻습니다.
//--- 그리고 저널에 제어 값을 표시합니다.
      value+=0.1;
      round_value=(int)MathRound(NormalizeDouble(value,1));
      PrintFormat("value: %.1f, round value: %d",value,round_value);
      /*
     결과:
      value: 0.1, round value: 0
      value: 0.2, round value: 0
      value: 0.3, round value: 0
      value: 0.4, round value: 0
      value: 0.5, round value: 1
      value: 0.6, round value: 1
```

© 2000-2025, MetaQuotes Ltd.

```
      value: 0.7, round value: 1
      value: 0.8, round value: 1
      value: 0.9, round value: 1
      value: 1.0, round value: 1
      value: 1.1, round value: 1
      value: 1.2, round value: 1
      value: 1.3, round value: 1
      value: 1.4, round value: 1
      value: 1.5, round value: 2
      value: 1.6, round value: 2
      value: 1.7, round value: 2
      value: 1.8, round value: 2
      value: 1.9, round value: 2
      value: 2.0, round value: 2
      value: 2.1, round value: 2
      value: 2.2, round value: 2
      value: 2.3, round value: 2
      value: 2.4, round value: 2
      value: 2.5, round value: 3
      value: 2.6, round value: 3
      value: 2.7, round value: 3
      value: 2.8, round value: 3
      value: 2.9, round value: 3
      value: 3.0, round value: 3
      value: 3.1, round value: 3
      */
   }
}
```

© 2000-2025, MetaQuotes Ltd.

MathSin

값이 가장 가까운 정수로 반올림됩니다.

```
double  MathSin(
   double  value      // 라디안 독립변수
   );
```

매개 변수

값

[in] 라디안 각도.

반환값

라디안으로 측정된 각도의 사인값. -1 ~1 범위의 값을 반환합니다.

참고

MathSin() 대신에 sin()을 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- 델타 단계를 사용하여 0에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 사인 값을 계산합니다.
   vector Y=MathSin(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathSqrt

숫자의 제곱근을 반환합니다.

```
double  MathSqrt(
   double  value      // 양수
   );
```

매개 변수

값

[in] 양수 값.

반환값

제곱근 값. 값이 음수이면 MathSqrt는 NaN(미확정 값)을 반환합니다.

참고

MathSqrt() 대신에 sqrt()를 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 1 스텝에서 0부터 10까지 11개의 값을 얻습니다.
   vector X(11,VectorArange);
   Print("vector X = \n",X);
//--- X 벡터의 각 값의 제곱근을 계산합니다.
   X=MathSqrt(X);
   Print("MathSqrt(X) = \n",X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double y_array[];
   X.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
   while(!IsStopped())
```

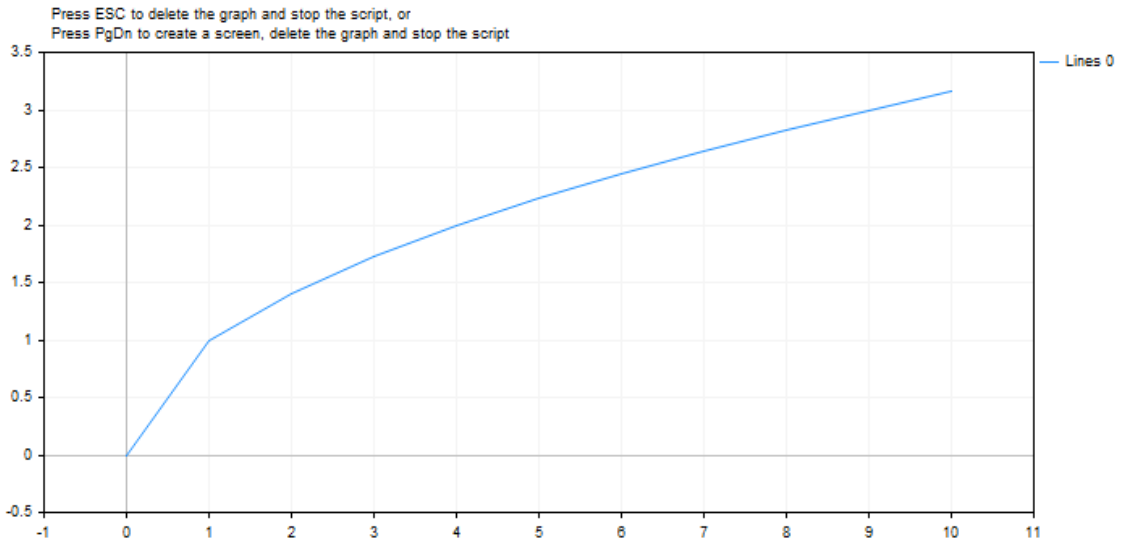
© 2000-2025, MetaQuotes Ltd.

```
      {
       if(StopKeyPressed())
          break;
       Sleep(16);
      }

//--- 모두 삭제
   ExtGraph.Destroy();
   /*
  결과:
   vector X =
   [0,1,2,3,4,5,6,7,8,9,10]
   MathSqrt(X) =
   [0,1,1.414213562373095,1.732050807568877,2,2.23606797749979,2.449489742783178,2.645
   */
  }
//+------------------------------------------------------------------+
//| '스텝' 증분 단위인 '값'으로 벡터 채우기                                  |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                      |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다                      |
//| 그렇지 않으면 'false'를 반환 합니다                                    |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                        |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                                |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



참고 항목

© 2000-2025, MetaQuotes Ltd.

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathSrand

일련의 무작위 랜덤 정수를 생성하기 위한 시작점을 설정합니다.

```
void  MathSrand(
   int  seed      // 시작 숫자
   );
```

Parameter

seed

[in] 난수 시퀀스의 시작 번호.

반환값

반환할 값이 없습니다.

참고

MathRand() 함수는 의사 난수 시퀀스를 생성하는 데 사용됩니다. 특정 초기화 숫자를 가진 MathSrand()를 호출하면 항상 동일한 유사 난수 시퀀스를 생성할 수 있습니다.

반복되지 않는 시퀀스의 수신을 보장하려면 MathSrand(GetTickCount())의 호출을 사용하십시오. GetTickCount()의 값은 운영 체제의 시작 순간부터 증가하고 밀리초 단위의 내장된 카운터가 오버플로될 때까지 49일 이내에 반복되지 않습니다. MathSrand(TimeCurrent())의 사용이 적합하지 않습니다. 왜냐하면 TimeCurrent() 함수가 가령 주말일 경우 오랫동안 변경되지 않을 수 있어, 마지막 확인된 시간을 반환합니다.

지표 및 Expert Advisor에 대해 MathSrand()를 사용하여 임의 번호 생성기를 초기화하는 것은 OnInit() 핸들러에서 더 잘 수행되며, OnTick() 및 OnCalculate()에서 생성기를 여러 번 다시 시작하는 것을 방지할 수 있습니다.

MathSrand() 함수 대신에 srand() 함수를 사용하실 수 있습니다.

예:

```
#property 설명 "지표는 중심 극한 정리를 나타내며, 이것은 다음과 같습니다:"
#property 설명 "약하게 종속된 랜덤 변수의 수가 충분히 많은 총합, "
#property 설명 "대략적으로 같은 크기를 가지고 있음(어떤 덧수도 지배하지 않음,"
#property 설명 "또는 합계에 대한 결정적 기여를 함)에 정규 분포에 가까운 분포가 있음."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- 그래픽 구조의 특성
#property indicator_label1  "레이블"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
//--- 입력 변수
input int      sample_number=10;
//--- 분포를 그리기 위한 지표 버퍼
double         LabelBuffer[];난수 생성기 초기화
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 틱 카운터
double         ticks_counter;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 배열 및 지표 버퍼 바인딩
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- 지표 버퍼를 현재에서 과거로 돌립니다
   ArraySetAsSeries(LabelBuffer,true);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 틱 카운터 초기화
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 영점 카운터 재설정시 지표 버퍼
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- 카운터 증가
   ticks_counter++;
//--- 주기적으로 카운터 눈금을 재설정해야 분포를 되살릴 수 있습니다.
   if(ticks_counter>100)
     {
      Print("지표 값을 재설정했습니다. 다시 한 번 셀 채우기를 시작하겠습니다.");
      ticks_counter=0;
     }
//--- 0에서 7까지의 세 숫자의 합으로 랜덤 값의 표본을 가져옵니다.
   for(int i=0;i<sample_number;i++)
     {
      //--- 셀의 지수 계산. 여기서 난수가 다른 세 개의 숫자의 합으로 됩니다.
      int rand_index=0;
      //--- 0에서 7까지 세 개의 난수를 구합니다.
      for(int k=0;k<3;k++)
        {
         //--- 7까지 나눗셈에 남은 값은 0부터 6까지의 값을 반환합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
         rand_index+=MathRand()%7;
        }
      //--- 셀 번호 rand_index의 값을 1만큼 늘립니다.
      LabelBuffer[rand_index]++;
     }
//--- OnCalculate() 핸들러 종료
   return(rates_total);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathTan

이 함수는 숫자의 탄젠트를 반환합니다.

```
double  MathTan(
   double  rad      // 라디안 독립변수
   );
```

매개 변수

rad

[in] 라디안 각도.

반환값

라디안 각의 탄젠트 값. 라디안 각이 263보다 크거나 같거나 -263보다 작거나 같으면 결과에서 유의성 상실이 발생하며, 이 경우 함수는 무한수를 반환합니다.

참고

MathTan() 대신 tan()을 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=-263;
//--- 델타 스텝을 사용하여 -263에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 탄젠트 값을 계산합니다.
   vector Y=MathTan(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);
```

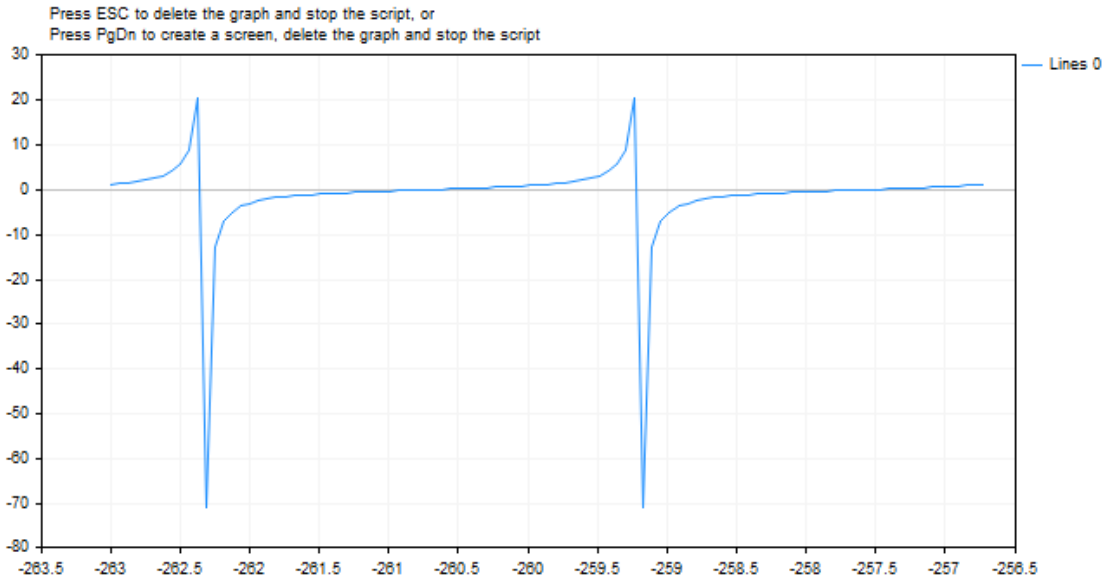
© 2000-2025, MetaQuotes Ltd.

```
//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRIT
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



참고 항목

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathIsValidNumber

이것은 실수의 정확성을 체크합니다.

```
bool  MathIsValidNumber(
   double  number      // 체크할 숫자
   );
```

Parameter

number

[in] 확인된 숫자 값.

반환값

확인된 값이 허용 가능한 실수일 경우 true를 반환합니다. 확인된 값이 +/- 무한대거나 "숫자가 아님"(NaN)이면 함수는 false를 반환합니다.

예:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Attention! MathArcsin(2.0) = ",abnormal);
```

참고 항목

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathExpm1

MathExp(x)-1 식의 값을 반환.

```
double  MathExpm1(
   double  value      // e의 거듭제곱
   );
```

매개 변수

값

[in] 거듭제곱을 명시하는 수.

반환값

double 유형의 값. 오버플로의 경우 함수는 INF (무한대)를 반환하고, 언더플로의 경우에는 MathExpm1은 0을 반환합니다.

참고

0에 가까운 **x**값에서 MathExpm1(x) 함수는 MathExp(x)-1 함수보다 훨씬 정확한 값을 생성합니다.

MathExpm1() 함수 대신에 expm1() 함수를 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 1 스텝에서 0부터 8까지 9개의 값을 얻습니다.
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- ("e"(오일러 수)를 각 X 벡터 값의 거듭제곱으로 계산) - 1
   X=MathExpm1(X);
   Print("MathExpm1(X) = \n",X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double y_array[];
   X.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(y_array,clrDodgerBlue);
```

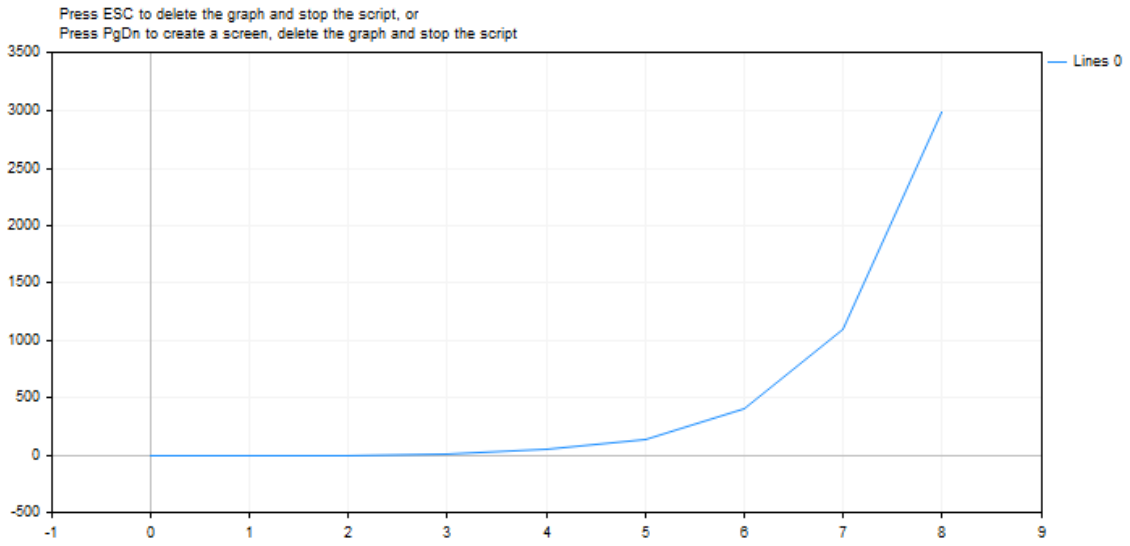
© 2000-2025, MetaQuotes Ltd.

```
//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| '스텝' 증분 단위인 '값'으로 벡터 채우기                                  |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                     |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다                    |
//| 그렇지 않으면 'false'를 반환 합니다                                   |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                       |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
```

© 2000-2025, MetaQuotes Ltd.

```
    ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
    ExtGraph.Update();
   }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                                |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
   {
    string file_names[];
    ResetLastError();
    int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
    if(selected<1)
      {
       if(selected<0)
          PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
       return false;
      }

    bool res=false;
    if(ChartSetInteger(0,CHART_SHOW,false))
       res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
    ChartSetInteger(0,CHART_SHOW,true);
    return(res);
   }
```

결과:



참고 항목

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathLog1p

MathLog(1+x) 식의 값을 반환합니다.

```
double  MathLog1p(
   double  value      // 로그를 씌울 값
   );
```

매개 변수

값

[in] 계산될 로그 값.

반환값

성공할 경우 값의 자연 로그(값 + 1)입니다. 값이 -1보다 작다면 함수는 NaN(정해지지않은 값)을 반환합니다. 값이 -1과 같으면 함수는 INF(무한대)를 반환합니다.

참고

x값이 0에 가까울 때 MathLog1p(x) 함수는 MathLog(1+x) 함수보다 훨씬 더 정확한 값을 생성합니다.

MathLog1p() 함수 대신에 log1p() 함수를 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 1 스텝에서 0부터 8까지 9개의 값을 얻습니다.
   vector X(9,VectorArange);
   Print("vector X = \n",X);
//--- 각 벡터 값에 대해 표현식의 로그를 계산합니다(1 + X 벡터 값).
   X=MathLog1p(X);
   Print("MathLog1p(X) = \n",X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double y_array[];
   X.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(y_array,clrDodgerBlue);
```

© 2000-2025, MetaQuotes Ltd.

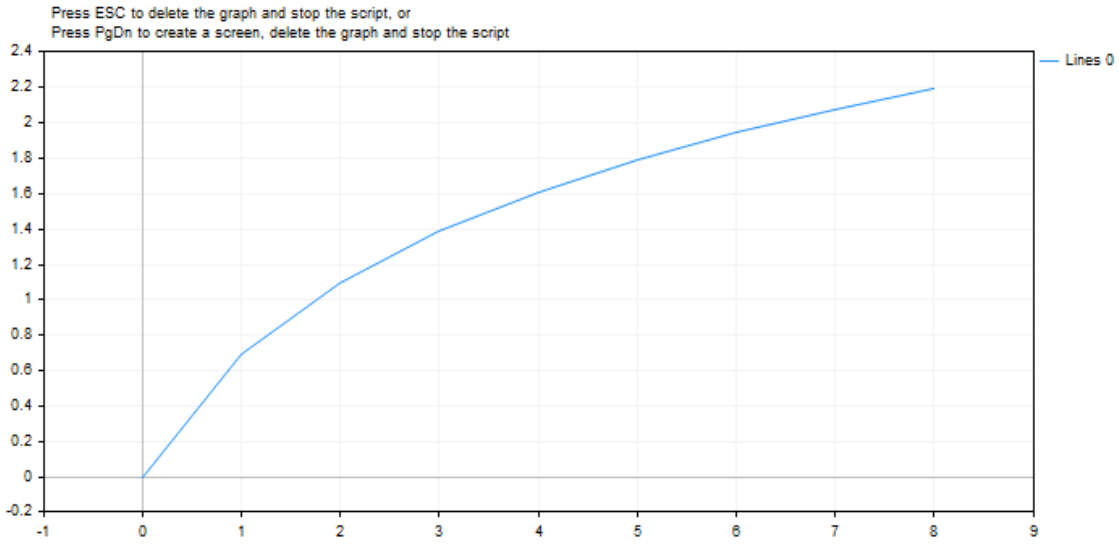
```
//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
   /*
  결과:
   vector X =
   [0,1,2,3,4,5,6,7,8]
   MathLog1p(X) =
   [0,0.6931471805599453,1.09861228866811,1.386294361119891,1.6094379124341,1.7917594
   */
  }
//+------------------------------------------------------------------+
//| '스텝' 증분 단위인 '값'으로 벡터 채우기                                  |
//+------------------------------------------------------------------+
template<typename T>
void VectorArange(vector<T> &vec,T value=0.0,T step=1.0)
  {
   for(ulong i=0; i<vec.Size(); i++,value+=step)
      vec[i]=value;
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                      |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다                      |
//| 그렇지 않으면 'false'를 반환 합니다                                    |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                        |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                                |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:

© 2000-2025, MetaQuotes Ltd.



참고 항목

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArccosh

쌍곡선 아크코사인을 반환

```
double  MathArccosh(
   double  value     // 1 <= 값 < ∞
   );
```

매개변수

값

[in] 계산되어야 하는 쌍곡선 아크코사인 값.

값 반환

수의 쌍곡선 아크코사인. 만약 값이 +1보다 작다면, 함수는 NaN (정의되지 않은 값)을 반환합니다.

주의

MathArccosh() 함수 대신에 acosh() 함수를 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/10);
   delta[0]=1;
//--- 델타 단계를 사용하여 1에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대해 쌍곡선 아크코사인 값을 계산합니다.
   vector Y=MathArccosh(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

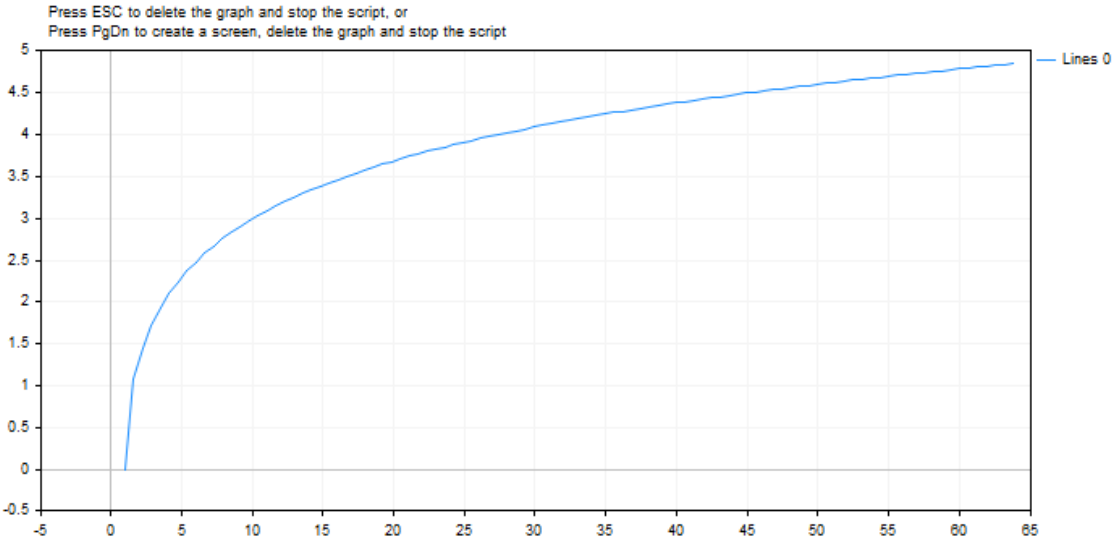
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



추가 참조

실제 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArcsinh

쌍곡선 아크사인을 반환.

```
double  MathArcsinh(
   double  value      // -∞ < 값 < +∞
   );
```

매개변수

val

[in] 계산되어야 하는 쌍곡선 아크사인 값

값 반환

수의 쌍곡선 아크사인.

주의

MathArcsinh() 함수 대신 asinh() 함수를 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/10);
   delta[0]=-31;
//--- 델타 단계를 사용하여 -31에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 쌍곡선 아크 사인 값을 계산합니다.
   vector Y=MathArcsinh(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

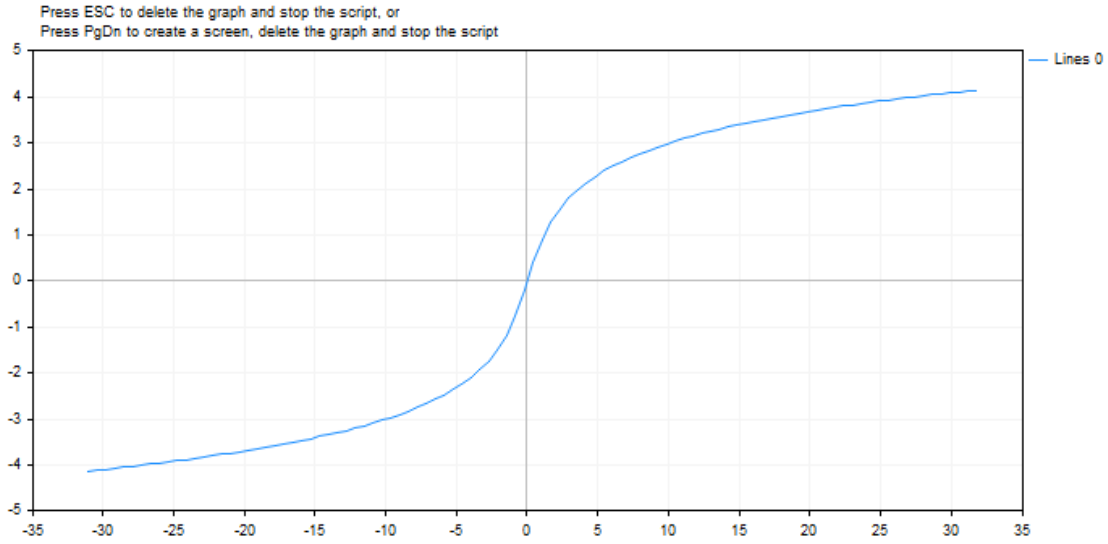
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



추가 참조

실제 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathArctanh

쌍곡선 아크탄젠트를 반환.

```
double  MathArctanh(
   double  value      // -1 < 값 < 1 범위 안의 값
   );
```

매개변수

값

[in] Number within the range of -1 < 값 < 1 범위 안에 있는 숫자로, 탄젠트를 나타냅니다.

값 반환

수의 쌍곡선 아크탄젠트.

주의

MathArctanh() 함수 대신 atanh() 함수를 사용할 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(34,2*M_PI/100);
   delta[0]=-1;
//--- 델타 단계를 사용하여 -1에서 2pi 까지 34개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 쌍곡선 탄젠트 값을 계산합니다.
   vector Y=MathArctanh(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

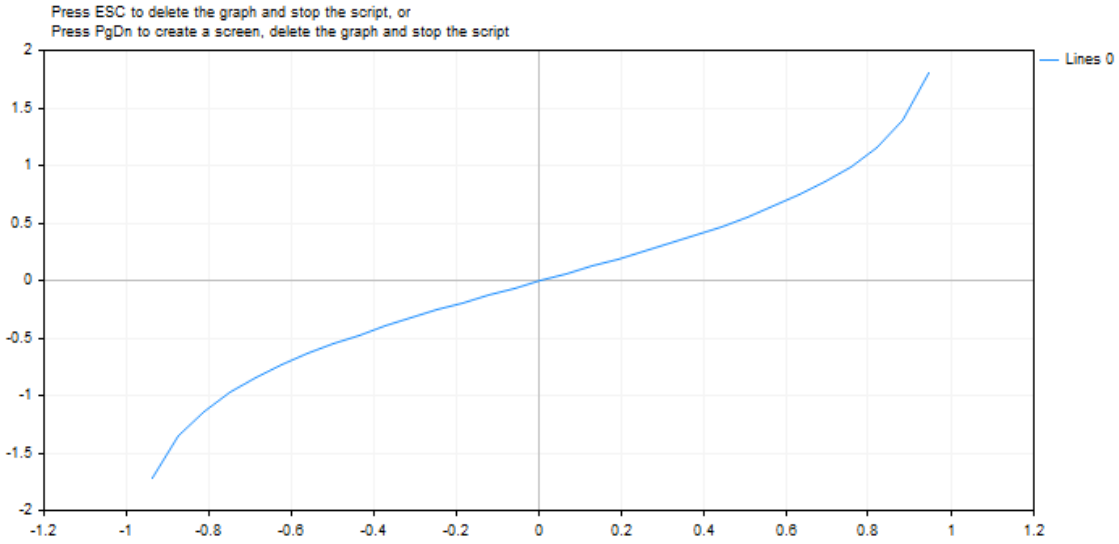
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



© 2000-2025, MetaQuotes Ltd.

MathCosh

숫자의 쌍곡선 코사인을 반환합니다.

```
double  MathCosh(
   double  value      // 숫자
   );
```

매개 변수

값

[in] 값.

반환값

숫자의 쌍곡선 코사인, +1 ~양의 무한대 범위 내의 값입니다.

참고

MathCosh() 함수 대신에 cosh() 함수를 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- 델타 단계를 사용하여 0에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 쌍곡선 코사인 값을 계산합니다.
   vector Y=MathCosh(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

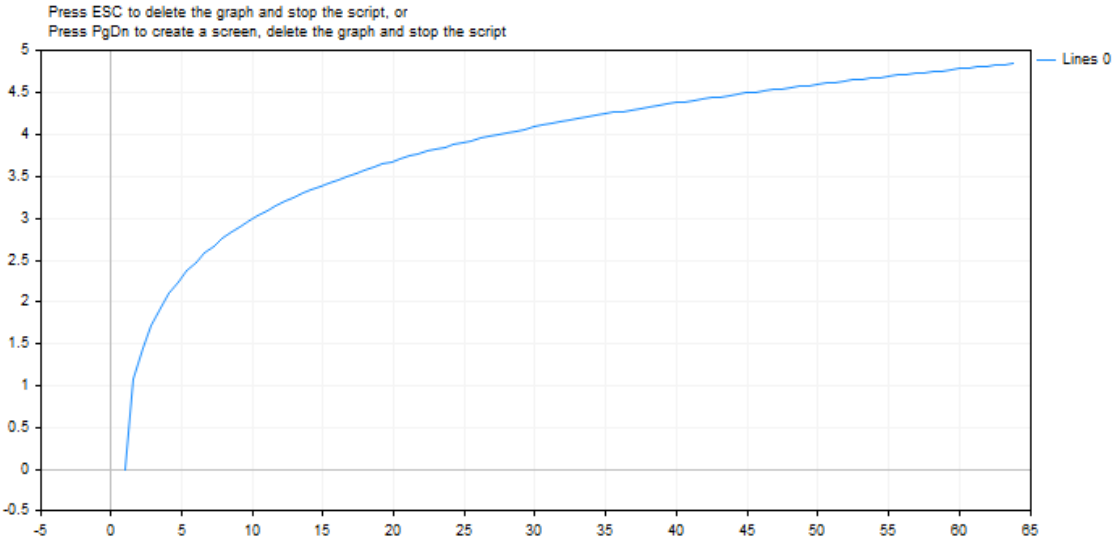
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



© 2000-2025, MetaQuotes Ltd.

MathSinh

숫자의 쌍곡선 사인을 반환합니다.

```
double  MathSinh(
   double  value      // 숫자
   );
```

매개 변수

값

[in] 값.

반환값

숫자의 쌍곡선 사인 값.

참고

MathSinh() 함수 대신에 sinh() 함수를 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- 델타 단계를 사용하여 0에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 쌍곡선 사인 값을 계산합니다.
   vector Y=MathSinh(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

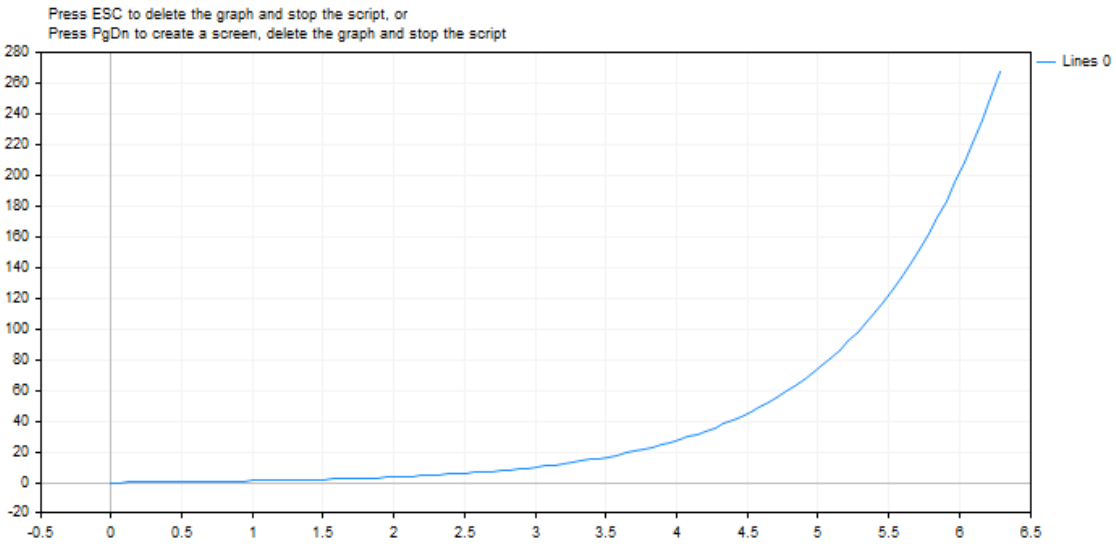
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



© 2000-2025, MetaQuotes Ltd.

MathTanh

숫자의 쌍곡선 탄젠트 값을 반환합니다.

```
double  MathTanh(
   double  value      // 숫자
   );
```

매개 변수

값

[in] 값.

반환값

숫자의 쌍곡선 탄젠트, -1 ~+1 범위 내의 값입니다.

참고

MathTanh() 함수 대신에 tanh() 함수를 사용하실 수 있습니다.

예:

```
#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

#include <Graphics\Graphic.mqh>

CGraphic ExtGraph;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector delta=vector::Full(101,2*M_PI/100);
   delta[0]=0;
//--- 델타 단계를 사용하여 0에서 2pi 까지 101개의 값을 얻습니다.
   vector X=delta.CumSum();
//--- X 벡터의 각 값에 대한 쌍곡선 탄젠트 값을 계산합니다.
   vector Y=MathTanh(X);

//--- 계산된 값을 벡터에서 배열로 전송합니다.
   double x_array[],y_array[];
   X.Swap(x_array);
   Y.Swap(y_array);

//--- 계산된 벡터 값의 그래프를 그립니다.
   CurvePlot(x_array,y_array,clrDodgerBlue);

//--- Escape 또는 PgDn 키를 눌러 그래프를 삭제하고(스크린샷 하기) 종료할 때까지 기다립니다.
```

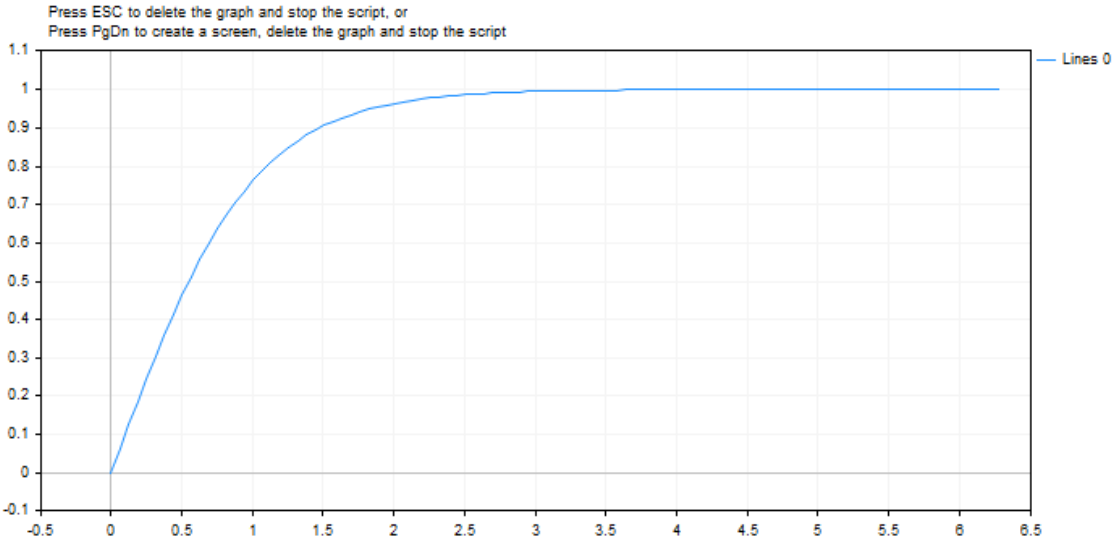
© 2000-2025, MetaQuotes Ltd.

```
   while(!IsStopped())
     {
      if(StopKeyPressed())
         break;
      Sleep(16);
     }

//--- 모두 삭제
   ExtGraph.Destroy();
  }
//+------------------------------------------------------------------+
//| ESC를 누르면 'true'를 반환합니다                                 |
//| PgDn을 누르면 그래프 스크린샷을 찍고 'true'를 반환합니다         |
//| 그렇지 않으면 'false'를 반환 합니다                              |
//+------------------------------------------------------------------+
bool StopKeyPressed()
  {
//--- ESC를 누르면 'true'를 반환합니다.
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
      return(true);
//--- PgDn을 누르고 그래프 스크린샷이 성공적으로 촬영되면 'true'를 반환합니다
   if(TerminalInfoInteger(TERMINAL_KEYSTATE_PAGEDOWN)!=0 && MakeAndSaveScreenshot(MQL
      return(true);
//--- 'false'를 반환합니다
   return(false);
  }
//+------------------------------------------------------------------+
//| 그래프 객체를 만들고 곡선 그리기                                 |
//+------------------------------------------------------------------+
void CurvePlot(double &x_array[], double &y_array[], const color colour)
  {
   ExtGraph.Create(ChartID(), "Graphic", 0, 0, 0, GRAPH_WIDTH, GRAPH_HEIGHT);
   ExtGraph.CurveAdd(x_array, y_array, ColorToARGB(colour), CURVE_LINES);
   ExtGraph.IndentUp(30);
   ExtGraph.CurvePlotAll();
stringtext1="그래프를 삭제하고 스크립트를 중지하려면 ESC를 누르십시오. 또는";
stringtext2=" 화면을 만들고 그래프를 삭제하고 스크립트를 중지하려면 PgDn을 누르세요";
   ExtGraph.TextAdd(54, 9, text1, ColorToARGB(clrBlack));
   ExtGraph.TextAdd(54,21, text2, ColorToARGB(clrBlack));
   ExtGraph.Update();
  }
//+------------------------------------------------------------------+
//| 스크린샷을 찍어 이미지를 파일로 저장                             |
//+------------------------------------------------------------------+
bool MakeAndSaveScreenshot(const string file_name)
  {
   string file_names[];
   ResetLastError();
   int selected=FileSelectDialog("Save Picture", NULL, "All files (*.*)|*.*", FSD_WRI
```

© 2000-2025, MetaQuotes Ltd.

```
   if(selected<1)
     {
      if(selected<0)
         PrintFormat("%s: FileSelectDialog() function returned error %d", __FUNCTION__
      return false;
     }

   bool res=false;
   if(ChartSetInteger(0,CHART_SHOW,false))
      res=ChartScreenShot(0, file_names[0], GRAPH_WIDTH, GRAPH_HEIGHT);
   ChartSetInteger(0,CHART_SHOW,true);
   return(res);
  }
```

결과:



참고 항목

Real 유형 (double, float)

© 2000-2025, MetaQuotes Ltd.

MathSwap

ushort 유형의 값에서 바이트 순으로 변경.

```
ushort  MathSwap(
   ushort  value      // 값
   );
```

매개 변수

값

[in] 바이트 순서를 변경하는 값.

반환값

역 바이트 순서의 ushort 값.

MathSwap

uint 유형 값에서 바이트 순서 변경.

```
uint  MathSwap(
   uint   value      // 값
   );
```

매개 변수

값

[in] 바이트 순서를 변경하는 값.

반환값

역 바이트 순서로의 uint 값.

MathSwap

ulong 유형 값에서 바이트 순서 변경.

```
ulong  MathSwap(
   ulong  value      // 값
   );
```

매개 변수

값

[in] 바이트 순서를 변경하는 값.

반환값

역 바이트 순서의 ulong 값.

예:

© 2000-2025, MetaQuotes Ltd.

```
#property script_show_inputs

input ulong  InpLongValue  =  1;    // 여기에 ulong 값을 입력
input uint   InpIntValue   =  2;    // 여기에 단위 값을 입력
input ushort InpShortValue =  3;    // 여기에 ushort 값을 입력

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- MathSwap()으로 입력하고 변환한 값을 저널에 10진수와 2진수로 출력합니다.
   Print(ValueDescription(InpLongValue));
   Print(ValueDescription(InpIntValue));
   Print(ValueDescription(InpShortValue));
   /*
  결과:
   ulong value: 1
   ulong value: 72057594037927936 using MathSwap()
   binary ulong value: 0000000000000000000000000000000000000000000000000000000000000000
   binary ulong value: 0000000100000000000000000000000000000000000000000000000000000000

   uint value: 2
   uint value: 33554432 using MathSwap()
   binary uint value: 00000000000000000000000000000010
   binary uint value: 00000010000000000000000000000000 using MathSwap()

   ushort value: 3
   ushort value: 768 using MathSwap()
   binary ushort value: 0000000000000011
   binary ushort value: 0000001100000000 using MathSwap()
   */
  }
//+------------------------------------------------------------------+
//| 변수 값을 설명하는 텍스트를 반환합니다.                                    |
//+------------------------------------------------------------------+
template <typename T>
string ValueDescription(T x)
  {
   int    num_bits   = sizeof(T)*8;
   string type_name  = typename(T);
   string bin_x      = NumberToBinaryString(x);
   string bin_swap_x = NumberToBinaryString(MathSwap(x));
   return(StringFormat("%s value: %lld\n%s value: %lld using MathSwap()\nbinary %s va
  }
//+------------------------------------------------------------------+
//| 숫자의 이진 표현법을 문자열로 반환                                        |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
template <typename T>
string NumberToBinaryString(T x)
  {
   string res  = "";
   int    i    = -1;
   uchar  size = sizeof(T)*8-1;
   ulong  mask = (ulong)1<<size;
   while(!((x<<++i)& mask));
   for(; i <=size; i++)
      res += !((x<<i)& mask) ? "0" : "1";
   return res;
  }
```

참고 항목

Network functions, SocketRead, SocketSend, SocketTlsRead, SocketTlsReadAvailable, SocketTlsSend

© 2000-2025, MetaQuotes Ltd.

문자열 함수

String 유형의 데이터로 작업하기 위한 함수 그룹입니다.

함수	기능
StringAdd	다른 문자열 끝에 문자열을 추가합니다.
StringBufferLen	문자열에 할당된 버퍼 크기를 반환합니다
StringCompare	두 문자열을 비교하여 첫 번째 문자열이 두 번째 문자열보다 크면 1을 반환하고, 문자열이 같은 경우 0을, 첫 번째 문자열이 두 번째 문자열보다 작으면 -1을 반환합니다
StringConcatenate	전달된 매개 변수 문자열을 형성합니다
StringFill	지정된 문자열을 선택한 심볼로 채웁니다
StringFind	문자열에서 하위 문자열 검색
StringGetCharacter	지정한 문자열 위치에 있는 숫자의 값을 반환합니다
StringInit	지정된 심볼로 문자열을 초기화하고 지정된 문자열 길이를 제공합니다
StringLen	문자열의 기호 수를 반환합니다
StringSetLength	문자열의 지정된 길이(글자로)를 설정합니다
StringReplace	문자열에서 발견된 모든 하위 문자열을 일련의 심볼로 바꿉니다
StringReserve	메모리에 있는 문자열에 대해 지정된 크기의 버퍼를 예약합니다.
StringSetCharacter	지정된 위치에 있는 심볼 값이 변경된 문자열의 복사본을 반환합니다
StringSplit	지정한 문자열에서 지정된 구분 심볼을 사용하여 하위 문자열을 가져오고 얻은 하위 문자열 수를 반환합니다
StringSubstr	지정한 위치에서 시작하는 텍스트 문자열에서 하위 문자열을 추출합니다
StringToLower	선택한 문자열의 모든 심볼을 소문자로 변환합니다
StringToUpper	선택한 문자열의 모든 심볼을 대문자로 변환합니다.
StringTrimLeft	문자열의 왼쪽 부분에 있는 줄 바꿈 문자, 공백 및 탭 잘라내기
StringTrimRight	문자열의 오른쪽에서 줄 바꿈 문자, 공백 및 탭 잘라내기

© 2000-2025, MetaQuotes Ltd.

StringAdd

이 함수는 문자열 끝에 하위 문자열을추가합니다.

```
bool  StringAdd(
   string&  string_var,        // 추가할 문자열
   string   add_substring      // 추가할 문자열
   );
```

Parameter

string_var

[in][out] 추가할 다른 문자열.

add_substring

[in] 소스 문자열 끝에 추가할 문자열.

반환값

성공 시 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류 코드를 얻으려면, GetLastError() 함수를 호출해야 합니다.

예:

```
void OnStart()
  {
   long length=1000000;
   string a="a",b="b",c;
//--- 첫번째 메소드
   uint start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);

//--- 두번째 메소드
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

//--- 세번째 메소드
   start=GetTickCount();
   a="a"; // re-initialize variable a
   for(i=0;i<length;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       StringConcatenate(c,a,b);
      }
   stop=GetTickCount();
   Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start)," milliseconds, i = ",i
  }
```

참고 항목

StringConcatenate, StringSplit, StringSubstr

© 2000-2025, MetaQuotes Ltd.

StringBufferLen

함수는 문자열에 할당된 버퍼 크기를 반환합니다.

```
int  StringBufferLen(
   string  string_var      // string
   )
```

Parameter

string_var

[in] 문자열.

반환값

0 값은 문자열이 일정하며 버퍼 크기를 변경할 수 없음을 의미합니다. -1은 문자열이 클라이언트 터미널에 속하며 버퍼 내용을 수정하면 불확실한 결과가 발생할 수 있습니다.

예:

```
void OnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("before: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("after: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
  }
```

참고 항목

StringAdd, StringInit, StringLen, StringFill

© 2000-2025, MetaQuotes Ltd.

StringCompare

함수는 두 문자열을 비교하고 비교 결과를 정수 형식으로 반환합니다.

```
int  StringCompare(
   const string&  string1,                  // 비교에서의 첫째 줄
   const string&  string2,                  // 비교에서의 둘째 줄
   bool           case_sensitive=true       // 비교를 위한 대/소문자 구분 모드 선택
   );
```

Parameter

string1

[in] 첫번째 문자열.

string2

[in] 두번째 문자열.

case_sensitive=true

[in] 대소문자 모드 선택. true라면, "A">"a" 입니다. false라면, "A"="a" 입니다. 기본적으로 값은 true와 같습니다.

반환값

- string1<string2 이라면 -1
- string1=string2 이라면 0
- string1>string2 이라면 1

참고

문자열은 심볼별로 비교되고 심볼은 현재 코드 페이지에 따라 알파벳 순서로 비교됩니다.

예:

```
void OnStart()
  {
//--- apple 과 home 중 어떤 것이 큽니까?
   string s1="Apple";
   string s2="home";

//--- 대소문자 비교
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("대소문자 비교: %s > %s",s1,s2);
   else
     {
      if(result1<0)PrintFormat("대소문자 비교: %s < %s",s1,s2);
      else PrintFormat("대소문자 비교: %s = %s",s1,s2);
     }

//--- 대소문자 비교
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("대소문자 비교: %s > %s",s1,s2);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       if(result2<0)PrintFormat("대소문자 비교: %s < %s",s1,s2);
       else PrintFormat("대소문자 비교: %s = %s",s1,s2);
      }
/* 결과:
     대소문자 비교: Apple < home
     대소문자 비교: Apple < home
*/
  }
```

참고 항목

String Type, CharToString(), ShortToString(), StringToCharArray(), StringToShortArray(), StringGetCharacter(), Use of a Codepage

© 2000-2025, MetaQuotes Ltd.

StringConcatenate

함수는 전달된 매개 변수의 문자열을 형성하고 형성된 문자열의 크기를 반환합니다. 매개 변수는 어떤 유형도 될 수 있습니다. 매개 변수의 수는 2보다 작거나 또는 64보다 클 수 없습니다.

```
int  StringConcatenate(
   string&  string_var,   // 결합할 문자열
   void argument1         // 모든 단일 유형의 첫 번째 매개 변수
   void argument2         // 모든 단일 유형의 두 번째 매개 변수
   ...                    // 모든 단일 유형의 다음 매개 변수
   );
```

매개 변수

string_var

[out] 연결의 결과로 형성되는 문자열.

argumentN

[in] 쉼표로 구분된 값. 모든 단순 유형의 2 -63개 매개 변수.

반환값

문자열 유형으로 변환된 매개 변수의 연결로 형성된 문자열 길이를 반환합니다. 매개변수는 Print() 및 Comment()에서와 동일한 규칙에 따라 문자열로 변환됩니다.

예:

```
void OnStart()
  {
//--- 결합에 참여하는 변수를 선언하고 정의합니다.
   string text="";
   string text1="This script shows how the StringConcatenate() function works.\n";
   string text2="This is the second line, at the end of which there is a line break co
   string text3="This is line number ";
   int    num3=3;
   string text31=", the number of which is entered into the function as a separate pa
   string textN="\n";
   string text4="This is line number 4, preceded by a separate parameter with a line b
   int    length=StringConcatenate(text, text1, text2, text3, num3, text31, textN, tex
   Print(text, "\nLength of the resulting string = ", length);

  /*
  결과
   This script shows how the StringConcatenate() function works.
   This is the second line, at the end of which there is a line break control code.
   This is line number 3, the number of which is entered into the function as a separa
   This is line number 4, preceded by a separate parameter with a line break code.
   Line 5 includes a real number: 0.12345
   Length of the resulting string = 358
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

참고 항목

StringAdd, StringSplit, StringSubstr

© 2000-2025, MetaQuotes Ltd.

StringFill

선택한 문자열을 지정된 기호별로 채웁니다.

```
bool  StringFill(
   string&   string_var,       // 채울 문자열
   ushort    character         // 문자열을 채울 심볼
   );
```

Parameter

string_var

[in][out] 선택한 심볼로 채워질 문자열.

character

[in] 문자열이 채워질 심볼.

반환값

성공 시 true, 그렇지 않으면 false. 에러 코드를 받으시려면 GetLastError()를 호출하십시오.

참고

문자열을 제자리에서 채우는 것은 새로운 문자열을 만들거나 복사하는 과도한 작업 없이 심볼이 문자열에 직접 삽입된다는 것을 의미합니다. 이렇게 하면 작업 시간을 절약할 수 있습니다.

예:

```
void OnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// 결과
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

참고 항목

StringBufferLen, StringLen, StringInit

© 2000-2025, MetaQuotes Ltd.

StringFind

문자열에서 부분 문자열을 검색합니다.

```
int  StringFind(
   string  string_value,        // 검색할 문자열
   string  match_substring,     // 검색된 것
   int     start_pos=0          // 어떤 위치에서부터 검색을 시작하는지.
   );
```

매개 변수

string_value

[in] 검색을 할 문자열.

match_substring

[in] 검색된 하위 문자열.

start_pos=0

[in] 검색을 시작할 문자열의 위치.

반환값

검색된 하위 문자열이 시작되는 문자열의 위치 번호 또는 하위 문자열을 찾을 수 없는 경우 -1을 반환합니다.

예:

```
#define   RESERVE    100

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 심볼 기본 통화와 수익 통화를 가져옵니다.
   string symbol_currency_base  =SymbolInfoString(Symbol(), SYMBOL_CURRENCY_BASE);
   string symbol_currency_profit=SymbolInfoString(Symbol(), SYMBOL_CURRENCY_PROFIT);
   PrintFormat("Symbol Currency Base: %s\nSymbol Currency Profit: %s", symbol_currency

//--- 서버에서 사용 가능한 모든 심볼의 루프
   int total=SymbolsTotal(false), pos=-1;
   for(int i=0; i<total; i++)
     {
//--- 다음 심볼의 이름을 얻습니다.
      string name=SymbolName(i, false);

//--- 기본 통화 이름이 포함된 심볼 이름에서 하위 문자열을 찾고
//--- 하위 문자열이 발견되면 심볼 이름, 해당 색인을 통화 목록에 표시하고 검색된 통화 이름을 로그
      pos = StringFind(name, symbol_currency_base);
      if(pos >= 0)
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("The '%s' symbol at index %u in the list contains the '%s' currer

//--- 인용된 통화명이 포함된 심볼명에서 하위 문자열을 찾고
//--- 하위 문자열이 발견되면 심볼 이름, 해당 색인을 통화 목록에 표시하고 검색된 통화 이름을 로그
      pos = StringFind(name, symbol_currency_profit);
      if(pos >= 0)
         PrintFormat("The '%s' symbol at index %u in the list contains the '%s' currer
     }

  /*
  결과
   StringFind (EURUSD,D1)   Symbol Currency Base: EUR
   StringFind (EURUSD,D1)   Symbol Currency Profit: USD
   The 'EURUSD' symbol at index 0 in the list contains the 'EUR' currency. Substring p
   The 'EURUSD' symbol at index 0 in the list contains the 'USD' currency. Substring p
   The 'GBPUSD' symbol at index 1 in the list contains the 'USD' currency. Substring p
   The 'USDCHF' symbol at index 2 in the list contains the 'USD' currency. Substring p
   The 'USDJPY' symbol at index 3 in the list contains the 'USD' currency. Substring p
   The 'USDCNH' symbol at index 4 in the list contains the 'USD' currency. Substring p
   The 'USDRUB' symbol at index 5 in the list contains the 'USD' currency. Substring p
   The 'AUDUSD' symbol at index 6 in the list contains the 'USD' currency. Substring p
   The 'NZDUSD' symbol at index 7 in the list contains the 'USD' currency. Substring p
   The 'USDCAD' symbol at index 8 in the list contains the 'USD' currency. Substring p
   The 'USDSEK' symbol at index 9 in the list contains the 'USD' currency. Substring p
   The 'USDHKD' symbol at index 10 in the list contains the 'USD' currency. Substring
   The 'USDSGD' symbol at index 11 in the list contains the 'USD' currency. Substring
   The 'USDNOK' symbol at index 12 in the list contains the 'USD' currency. Substring
   The 'USDDKK' symbol at index 13 in the list contains the 'USD' currency. Substring
   The 'USDTRY' symbol at index 14 in the list contains the 'USD' currency. Substring
   The 'USDZAR' symbol at index 15 in the list contains the 'USD' currency. Substring
   The 'USDCZK' symbol at index 16 in the list contains the 'USD' currency. Substring
   The 'USDHUF' symbol at index 17 in the list contains the 'USD' currency. Substring
   The 'USDPLN' symbol at index 18 in the list contains the 'USD' currency. Substring
   The 'USDRUR' symbol at index 19 in the list contains the 'USD' currency. Substring
   The 'EURAUD' symbol at index 27 in the list contains the 'EUR' currency. Substring
   The 'EURCAD' symbol at index 28 in the list contains the 'EUR' currency. Substring
   The 'EURCHF' symbol at index 29 in the list contains the 'EUR' currency. Substring
   The 'EURCZK' symbol at index 30 in the list contains the 'EUR' currency. Substring
   The 'EURDKK' symbol at index 31 in the list contains the 'EUR' currency. Substring
   The 'EURGBP' symbol at index 32 in the list contains the 'EUR' currency. Substring
   The 'EURHKD' symbol at index 33 in the list contains the 'EUR' currency. Substring
   The 'EURHUF' symbol at index 34 in the list contains the 'EUR' currency. Substring
   The 'EURJPY' symbol at index 35 in the list contains the 'EUR' currency. Substring
   The 'EURNOK' symbol at index 36 in the list contains the 'EUR' currency. Substring
   The 'EURNZD' symbol at index 37 in the list contains the 'EUR' currency. Substring
   The 'EURPLN' symbol at index 38 in the list contains the 'EUR' currency. Substring
   The 'EURRUR' symbol at index 39 in the list contains the 'EUR' currency. Substring
   The 'EURRUB' symbol at index 40 in the list contains the 'EUR' currency. Substring
   The 'EURSEK' symbol at index 41 in the list contains the 'EUR' currency. Substring
```

© 2000-2025, MetaQuotes Ltd.

```
   The 'EURTRY' symbol at index 42 in the list contains the 'EUR' currency. Substring
   The 'EURZAR' symbol at index 43 in the list contains the 'EUR' currency. Substring
   The 'XAUUSD' symbol at index 47 in the list contains the 'USD' currency. Substring
   The 'XAUEUR' symbol at index 48 in the list contains the 'EUR' currency. Substring
   The 'XAGUSD' symbol at index 50 in the list contains the 'USD' currency. Substring
   The 'XAGEUR' symbol at index 51 in the list contains the 'EUR' currency. Substring
   The 'USDCRE' symbol at index 53 in the list contains the 'USD' currency. Substring
   The 'XPDUSD' symbol at index 65 in the list contains the 'USD' currency. Substring
   The 'XPTUSD' symbol at index 66 in the list contains the 'USD' currency. Substring
   The 'USDGEL' symbol at index 67 in the list contains the 'USD' currency. Substring
   The 'USDMXN' symbol at index 68 in the list contains the 'USD' currency. Substring
   The 'EURMXN' symbol at index 69 in the list contains the 'EUR' currency. Substring
   The 'USDCOP' symbol at index 75 in the list contains the 'USD' currency. Substring
   The 'USDARS' symbol at index 76 in the list contains the 'USD' currency. Substring
   The 'USDCLP' symbol at index 77 in the list contains the 'USD' currency. Substring
   The 'EURSGD' symbol at index 89 in the list contains the 'EUR' currency. Substring
   The 'USDILS' symbol at index 95 in the list contains the 'USD' currency. Substring
   The 'USDTHB' symbol at index 122 in the list contains the 'USD' currency. Substring
   The 'USDRMB' symbol at index 123 in the list contains the 'USD' currency. Substring
   The 'EURILS' symbol at index 126 in the list contains the 'EUR' currency. Substring
   The 'EURCNH' symbol at index 137 in the list contains the 'EUR' currency. Substring
   The 'USDBRL' symbol at index 139 in the list contains the 'USD' currency. Substring
  */
  }
```

참고 항목

StringSubstr, StringGetCharacter, StringLen, StringLen

© 2000-2025, MetaQuotes Ltd.

StringGetCharacter

문자열의 지정된 위치에 있는 심볼 값을 반환합니다.

```
ushort  StringGetCharacter(
   string  string_value,     // 문자열
   int     pos               // 문자열에서의 심볼 위치
   );
```

매개 변수

string_value

[in] 문자열.

pos

[in] 문자열에서의 심볼 위치. 0에서 StringLen(text) -1 사이일 수 있습니다.

반환값

오류 발생 시 심볼 코드 또는 0. 에러 코드를 받으시려면 GetLastError()를 호출하십시오.

예:

```
void OnStart()
  {
//--- 차트에 있는 모든 코멘트를 삭제합니다.
   Comment("");
//--- 심볼 코드의 값을 얻고 문자열 길이를 기억하는 문자열을 선언합니다.
   string message = "The script demonstrates the operation of the StringGetCharacter()
   int    length  = StringLen(message);
//--- 데모 문자열에서 얻은 심볼을 추가할 문자열 변수를 선언합니다.
   string text    = "";
//--- 데모 문자열 길이만큼 루프에서
   for(int i=0; i<length; i++)
     {
//--- 1/10초 기다립니다
      Sleep(100);
//--- 데모 문자열의 루프 인덱스에 있는 문자열에서 심볼을 가져옵니다.
      ushort char_code=StringGetCharacter(message, i);
//--- 표시된 문자열에 심볼을 추가하고 결과 문자열을 차트 코멘트로 표시합니다.
      text+=ShortToString(char_code);
      Comment(text);
     }
//--- 2초 정도 기다린 후 차트에서 주석을 제거합니다.
   Sleep(2000);
   Comment("");

  /*
  결과: 데모 문자열이 화면에 문자별로 표시됩니다.
   The script demonstrates the operation of the StringGetCharacter() function
  */
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

참고 항목

StringSetCharacter, StringBufferLen, StringLen, StringFill, StringInit, StringToCharArray, StringToShortArray

© 2000-2025, MetaQuotes Ltd.

StringInit

지정된 심볼별로 문자열을 초기화하고 지정된 문자열 크기를 제공합니다.

```
bool  StringInit(
   string&   string_var,        // 초기화할 문자열
   int       new_len=0,         // 초기화 후 필요한 문자열 길이
   ushort    character=0        // 문자열이 채워질 심볼
   );
```

Parameter

string_var

[in][out] 초기화 및 초기화 취소해야 하는 문자열.

new_len=0

[in] 초기화 후 문자열 길이. 길이=0이면 문자열을 초기화 해제하는데, 즉, 문자열 버퍼가 지워지고 버퍼 주소가 0이 됩니다.

character=0

[in] 문자열을 채울 심볼.

반환값

성공 시 true, 그렇지 않으면 false. 에러 코드를 받으시려면 GetLastError()를 호출하십시오.

참고

문자 =0과 길이가 new_len>0인 경우, 표시된 길이의 문자열의 버퍼가 배포되고 0으로 채워집니다. 전체 버퍼가 문자열 종결자에 의해 채워지기 때문에 문자열 길이는 0과 같습니다.

예:

```
void OnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/*  결과
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

참고 항목

StringBufferLen, StringLen

© 2000-2025, MetaQuotes Ltd.

StringLen

문자열의 심볼 수를 반환합니다.

```
int  StringLen(
   string  string_value      // 문자열
   );
```

매개 변수

string_value

[in] 길이를 계산할 문자열.

반환값

끝 0이 없는 문자열의 심볼 수.

예:

```
void OnStart()
  {
//--- 테스트 문자열 정의
   string text="123456789012345";
//--- 문자열의 심볼의 수를 가져옵니다.
   int str_len=StringLen(text);
//--- 문자열과 그 안의 심볼 수를 로그에 표시합니다.
   PrintFormat("The StringLen() function returned the value of %d chars in string: '%s

  /*
  결과
   The StringLen() function returned the value of 15 chars in string: '12345678901234
  */
  }
```

참고 항목

StringBufferLen, StringTrimLeft, StringTrimRight, StringToCharArray, StringToShortArray

© 2000-2025, MetaQuotes Ltd.

StringSetLength

문자열의 지정된 길이(문자)를 설정합니다.

```
bool  StringSetLength(
   string&     string_var,       // 문자열
   uint        new_length        // 새 문자열 길이
   );
```

매개 변수

string_var

[in][out] 새 길이를 문자 단위로 설정할 문자열.

new_capacity

[in] 필요한 문자열 길이(문자). new_length가 현재 크기보다 작으면 초과한 문자가 삭제됩니다.

반환값

실행이 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 에러 코드를 받으려면, GetLastError() 함수를 호출해야 합니다.

참고

StringSetLength() 함수는 문자열에 할당된 버퍼의 크기를 변경하지 않습니다.

예:

```
void OnStart()
  {
//--- 문자열 정의
   string text="123456789012345";

//--- 로그에 문자열과 그 길이를 표시합니다.
   PrintFormat("Before StringSetLength() the string '%s' has a size of %d characters",

//--- 문자열 크기를 10자로 줄입니다.
   StringSetLength(text, 10);

//--- StringSetLength() 작업으로 인해 변경된 문자열과 새로운 길이를 로그에 표시합니다.
   PrintFormat("After StringSetLength() the string is now '%s', and has a size of %d c

  /*
  결과
   Before StringSetLength() the string '123456789012345' has a size of 15 characters
   After StringSetLength() the string is now '1234567890', and has a size of 10 charac
  */
  }
```

참고 항목

StringLen, StringBufferLen, StringReserve StringInit, StringSetCharacter

© 2000-2025, MetaQuotes Ltd.

StringReplace

문자열에서 발견된 모든 하위 문자열을 일련의 심볼로 바꿉니다.

```
int  StringReplace(
   string&         str,              // 하위 문자열을 교체할 끈
   const string    find,             // 찾은 하위 문자열
   const string    replacement       // 검색된 위치에 삽입될 하위 문자열
   );
```

Parameter

str

[in][out] 하위 문자열을 교체할 문자열.

find

[in] 교체를 희망하는 하위 문자열.

replacement

[in] 찾은 문자열 대신 삽입할 문자열.

반환값

이 함수는 성공 시 교체 횟수를 반환하고, 그렇지 않으면 -1입니다. 에러 코드를 받으시려면 GetLastError() 함수를 호출하십시오.

참고

함수가 성공적으로 실행되었지만 교체가 이루어지지 않은 경우(교체할 하위 문자열을 찾을 수 없음) 0을 반환합니다.

이 오류는 잘못된 *str* 또는 *find* 매개 변수 (빈 문자열 또는 초기화되지 않은 문자열, StringInit() 참조)로 인해 발생할 수 있습니다. 또한 메모리가 부족하여 교체를 완료할 수 없을 경우 오류가 발생합니다.

예:

```
  string text="The quick brown fox jumped over the lazy dog.";
  int replaced=StringReplace(text,"quick","slow");
  replaced+=StringReplace(text,"brown","black");
  replaced+=StringReplace(text,"fox","bear");
  Print("Replaced: ", replaced,". Result=",text);

//  결과
//  교체됨: 3. Result=The slow black bear jumped over the lazy dog.
//
```

참고 항목

StringSetCharacter(), StringSubstr()

© 2000-2025, MetaQuotes Ltd.

StringReserve

메모리에 있는 문자열에 대해 지정된 크기의 버퍼를 예약합니다.

```
bool  StringReserve(
   string&     string_var,       // 문자열
   uint        new_capacity      // 문자열을 저장하기 위한 버퍼 크기
   );
```

Parameter

string_var

[in][out] 버퍼 크기를 변경해야 하는 문자열.

new_capacity

[in] 문자열에 필요한 버퍼 크기. new_capacity의 크기가 문자열 길이보다 작으면 현재 버퍼의 크기는 변경되지 않습니다.

반환값

실행이 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 에러 코드를 받으려면, GetLastError() 함수를 호출해야 합니다.

참고

일반적으로 문자열 크기는 문자열을 저장하기 위한 버퍼 크기와 같지 않습니다. 문자열을 생성할 때 일반적으로 적절한 버퍼가 여백과 함께 할당됩니다. StringReserve() 함수를 사용하면 버퍼 크기를 관리할 수 있으며 이후 작업에 적합한 크기를 지정할 수 있습니다.

StringInit() 와는 달리, StringReserve() 함수는 문자열 내용을 변경하지 않고 문자로 채우지 않습니다.

예:

```
void OnStart()
  {
   string s;
//--- StringReserve를 사용하지 않고 작업 속도를 확인합니다
   ulong t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_no_reserve=GetMicrosecondCount()-t0;
   s=NULL;
//--- 이제 StringReserve를 사용하여 동일한 작업을 수행하겠습니다
   StringReserve(s,1024 * 3);
   t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_reserve=GetMicrosecondCount()-t0;
//--- 시간 체크
   Print("Test with StringReserve passed for "+(string)msc_reserve+" msc");
   Print("Test without StringReserve passed for "+(string)msc_no_reserve+" msc");
/* 결과:
    50msc 동안 통과된 StringReserve를 사용한 테스트
```

© 2000-2025, MetaQuotes Ltd.

```
    121 msc 동안 통과된 StringReserve 없이 한 테스트
*/
  }
```

참고 항목

StringBufferLen, StringSetLength, StringInit, StringSetCharacter

© 2000-2025, MetaQuotes Ltd.

StringSetCharacter

지정된 위치에 변경된 문자가 있는 문자열의 복사본을 반환합니다.

```
bool  StringSetCharacter(
   string&   string_var,       // 문자열
   int       pos,              // 위치
   ushort    character         // 글자
   );
```

Parameter

string_var

[in][out] 문자열.

pos

[in] 문자열에서의 글자의 위치. 0에서 StringLen(text) 까지 가능.

character

[in] 기호 코드 유니코드.

반환값

성공 시 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류 코드를 얻으려면, GetLastError() 함수를 호출해야 합니다.

참고

pos가문자열 길이보다 작고 기호 코드 값이 0이면 문자열이 차단됩니다(그러나 버퍼 사이즈, 문자열에 대해 분산되어 있음). 문자열 길이가 pos와 같아집니다.

*pos*가 문자열 길이와 같으면 문자열 끝에 지정된 기호가 추가되고 길이가 1씩 확대됩니다.

예:

```
void OnStart()
  {
   string str="0123456789";
   Print("before: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- 가운데 0 값 추가
   StringSetCharacter(str,6,0);
   Print("after: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- 끝에 기호 추가
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("addition: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* 결과
   before: str = 0123456789 ,StringBufferLen(str) = 0   StringLen(str) = 10
    after: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
```

© 2000-2025, MetaQuotes Ltd.

```
   addition: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
*/
```

참고 항목

StringBufferLen, StringLen, StringFill, StringInit, CharToString, ShortToString, CharArrayToString, ShortArrayToString

© 2000-2025, MetaQuotes Ltd.

StringSplit

지정한 문자열에서 지정된 구분자로 하위 문자열을 가져오고 얻은 하위 문자열 수를 반환합니다.

```
int  StringSplit(
   const string   string_value,        // 검색할 문자열
   const ushort   separator,           // 하위 문자열을 검색할 구분 기호
   string         & result[]           // 발견된 하위 문자열을 가져오기 위해 참조로 전달된 배
   );
```

Parameter

string_value

[in] 하위 문자열을 가져와야 하는 문자열. 문자열은 변경되지 않습니다.

pos

[in] 구분 문자의 코드. 코드를 얻으려면 StringGetCharacter() 함수를 사용할 수 있습니다.

result[]

[out] 획득한 하위 문자열이 있는 문자열 배열.

반환값

result[] 배열의 하위 문자열 수. 전달된 문자열에 구분 기호가 없으면 하나의 소스 문자열만 배열에 배치됩니다.

string_value가 비어 있거나 NULL이면 함수는 0을 반환합니다. 오류가 발생하면 함수는 -1을 반환합니다. 에러 코드를 받으려시면, GetLastError() 함수를 호출하십시오.

예:

```
string to_split="_life_is_good_"; // 하위 문자열로 분할할 문자열
   string sep="_";                // 구분 기호 문자
   ushort u_sep;                  // 구분 문자의 코드
   string result[];               // 문자열을 가져올 배열
   //--- 구분 코드 획득
   u_sep=StringGetCharacter(sep,0);
   //--- 문자열을 하위 문자열로 분할
   int k=StringSplit(to_split,u_sep,result);
   //--- 코멘트 보이기
   PrintFormat("획득한 문자열: %d. 코드 %d에 대해 구분자 %s를 사용하십시오",k,sep,u_sep);
   //--- 이제 얻은 문자열을 모두 출력합니다.
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

참고 항목

StringReplace(), StringSubstr(), StringConcatenate()

© 2000-2025, MetaQuotes Ltd.

StringSubstr

지정된 위치에서 시작하는 텍스트 문자열에서 하위 문자열을 추출합니다.

```
string  StringSubstr(
   string  string_value,     // 문자열
   int     start_pos,        // 시작 위치
   int     length=-1         // 추출된 문자열 길이
   );
```

매개 변수

string_value

[in] 하위 문자열을 추출할 문자열.

start_pos

[in] 하위 문자열의 초기 위치. 0에서 StringLen(text) -1 사이일 수 있습니다.

length=-1

[in] 추출된 하위 문자열의 길이. 파라미터 값이 -1과 같거나 파라미터가 설정되지 않은 경우, 하위 문자열이 표시된 위치에서 문자열 끝까지 추출됩니다.

반환값

가능한 경우 추출된 하위 문자열의 복사본. 그렇지 않으면 빈 문자열을 반환합니다.

예:

```
void OnStart()
  {
//--- 현재 심볼명을 가져옵니다.
   string name = Symbol();

//--- 기본 및 쿼트된 심볼 통화를 가져옵니다.
   string base   = StringSubstr(name, 0, 3);
   string quoted = StringSubstr(name, 3, 3);

//--- 얻은 심볼 통화를 로그에 표시합니다.
   PrintFormat("Symbol: %s. Currency base: %s, currency quoted: %s", name, base, quote

  /*
  결과
   Symbol: EURUSD. Currency base: EUR, currency quoted: USD
  */
  }
```

참고 항목

StringSplit, StringFind, StringGetCharacter

© 2000-2025, MetaQuotes Ltd.

StringToLower

선택한 문자열의 모든 심볼을 소문자로 변환합니다.

```
bool  StringToLower(
   string&  string_var      // 처리할문자열
   );
```

매개 변수

string_var

[in][out] 문자열.

반환값

성공 시 true, 그렇지 않으면 false. 에러 코드를 받으시려면 GetLastError()를 호출하십시오.

예:

```
void OnStart()
  {
//--- 소스 문자열을 대문자로 정의합니다.
   string text=" - THIS STRING, WRITTEN IN UPPERCASE, MUST BE WRITTEN IN LOWERCASE";
//--- 로그에 소스 문자열을 표시합니다.
   Print("Source line:\n", text);
//--- 모든 문자열 문자를 소문자로 변환하고 결과를 로그에 표시합니다.
   if(StringToLower(text))
      Print("The original string after using the StringToLower() function:\n", text);

  /*
  결과
   Source line:
    - THIS STRING, WRITTEN IN UPPERCASE, MUST BE WRITTEN IN LOWERCASE
   The original string after using the StringToLower() function:
    - this string, written in uppercase, must be written in lowercase
  */
  }
```

참고 항목

StringToUpper, StringTrimLeft, StringTrimRight

© 2000-2025, MetaQuotes Ltd.

StringToUpper

선택한 문자열의 모든 기호를 대문자로 변환합니다.

```
bool  StringToUpper(
   string&  string_var      // 처리할문자열
   );
```

매개 변수

string_var

[in][out] 문자열.

반환값

성공 시 true, 그렇지 않으면 false. 에러 코드를 받으시려면 GetLastError()를 호출하십시오.

예:

```
void OnStart()
  {
//--- 소스 문자열을 소문자로 정의합니다.
   string text=" - this string, written in lowercase, must be written in uppercase";
//--- 로그에 소스 문자열을 표시합니다.
   Print("Source line:\n", text);
//--- 모든 문자열 문자를 대문자로 변환하고 결과를 로그에 표시합니다.
   if(StringToUpper(text))
      Print("The original string after using the StringToUpper() function:\n", text);

  /*
  결과
   Source line:
    - this string, written in lowercase, must be written in uppercase
   The original string after using the StringToUpper() function:
    - THIS STRING, WRITTEN IN LOWERCASE, MUST BE WRITTEN IN UPPERCASE
  */
  }
```

참고 항목

StringToLower, StringTrimLeft, StringTrimRight

© 2000-2025, MetaQuotes Ltd.

StringTrimLeft

이 함수는 문자열 왼쪽 부분의 줄바꿈 문자, 공백 및 탭을 첫 번째 의미 있는 기호까지 잘라냅니다. 여기서 문자열이 수정됩니다.

```
int  StringTrimLeft(
   string&  string_var      // 자를 문자열
   );
```

매개 변수

string_var

[in][out] 왼쪽에서 잘라낼 문자열.

반환값

자를 기호 수를 반환합니다.

예:

```
void OnStart()
  {
//--- 왼쪽에 6개의 공백을 사용하여 소스 문자열을 정의합니다.
   string text="      All spaces on the left will be removed from this string";
//--- 로그에 소스 문자열을 표시합니다.
   PrintFormat("Source line:\n'%s'", text);
//--- 왼쪽의 모든 공백을 제거하고 제거된 문자 수와 결과 문자열을 로그에 표시합니다.
   int num=StringTrimLeft(text);
   PrintFormat("The StringTrimLeft() function removed %d chars from the left side. Now

  /*
  결과
   Source line:
   '      All spaces on the left will be removed from this string'
   The StringTrimLeft() function removed 6 chars from the left side. Now the line look
   'All spaces on the left will be removed from this string'
  */
  }
```

참고 항목

StringTrimRight, StringToLower, StringToUpper

© 2000-2025, MetaQuotes Ltd.

StringTrimRight

이 함수는 문자열 오른쪽에서 마지막 의미 기호 뒤에 있는 줄 바꿈 문자, 공백 및 탭을 잘라냅니다. 여기서 문자열이 수정됩니다.

```
int  StringTrimRight(
   string&  string_var      // 자를 문자열
   );
```

매개 변수

string_var

[in][out] 오른쪽에서 잘라낼 문자열.

반환값

자를 기호 수를 반환합니다.

예:

```
void OnStart()
  {
//--- 오른쪽에 6개의 공백을 사용하여 소스 문자열을 정의합니다.
   string text="All spaces on the right will be removed from this string      ";
//--- 로그에 소스 문자열을 표시합니다.
   PrintFormat("Source line:\n'%s'", text);
//--- 오른쪽의 모든 공백을 제거하고 제거된 문자 수와 결과 문자열을 로그에 표시합니다.
   int num=StringTrimRight(text);
   PrintFormat("The StringTrimRight() function removed %d chars from the right side. N

   /*
  결과
   Source line:
   'All spaces on the right will be removed from this string      '
   The StringTrimRight() function removed 6 chars from the right side. Now the line lo
   'All spaces on the right will be removed from this string'
   */
  }
```

참고 항목

StringTrimLeft, StringToLower, StringToUpper

© 2000-2025, MetaQuotes Ltd.

날짜 및 시간

이는 datetime 유형의 데이터(1920년 1월 1일 0시부터 경과된 시간(초)을 나타내는 정수)로 작업하기 위한 기능 그룹입니다.

고해상도 카운터 및 타이머를 정렬하려면 밀리초 단위의 값을 생성하는 GetTickCount() 함수를 사용합니다.

기능	액션
TimeCurrent	알려진 마지막 서버 시간(마지막 시세 수령 시간)을 날짜시간 형식으로 반환
TimeTradeServer	거래 서버의 현재 게산 시간을 반환
TimeLocal	로컬 컴퓨터 시간을 날짜시간 형식으로 반환
TimeGMT	클라이언트 터미널이 실행 중인 컴퓨터 로컬 시간별 데이라이트 세이빙 타임으로 GMT 날짜시간 형식으로 반환
TimeDaylightSavings	데이라이트 세이빙 타임 스위치 신호를 반환
TimeGMTOffset	DST 스위치를 고려하여 GMT 시간과 로컬 컴퓨터 시간 사이의 현재 차이(초)를 반환
TimeToStruct	날짜시간 값을 MqlDateTime 구조 유형의 변수로 변환
StructToTime	MqlDateTime 구조 유형의 변수를 날짜시간 값으로 변환₩

© 2000-2025, MetaQuotes Ltd.

TimeCurrent

"마켓 워치(Market Watch)" 창에서 선택한 심볼 중 하나에 대해 마지막으로 확인된 서버 시간, 마지막 시세 수신 시간을 반환. OnTick() 처리기에서 이 기능은 수신된 틱 처리 시간을 반환. 다른 경우(예:핸들러 OnInit(), OnDeinit(), OnTimer()등에서 호출)는 "마켓 워치(Market Watch)" 창에서 사용할 수 있는 심볼에 대한 마지막 시세 수신 시간이며, 이 창 제목에 표시된 시간입니다. 시간 값은 거래 서버에서 형성되며 시스템의 시간 설정에 따라 달라지지 않습니다. 함수의 2가지 변수가 있습니다.

매개변수 없이 호출

```
datetime  TimeCurrent();
```

MqlDateTime 유형 매개변수를 사용하여 호출

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // 구조 유형 변수
   );
```

매개변수

dt_struct

[out] MqlDateTime 구조 유형 변수.

값 반환

datetime 유형의 값

주의

MqlDateTime 구조 유형 변수가 매개변수로 전달된 경우 그에 따라 채워집니다.

고해상도 카운터 및 타이머를 정렬하려면 밀리초 단위의 값을 생성하는 GetTickCount() 함수를 사용합니다.

전략 테스터에서 테스트하는 동안 과거 데이터에 따라 TimeCurrent() 이 시뮬레이션됩니다.

예:

```
void OnStart()
  {
//--- 날짜/시간 데이터로 채워질 MqlDateTime 변수를 선언하고 Market Watch 창에서 마지막 시세 시
   MqlDateTime tm={};
   datetime    time1=TimeCurrent();    // 첫 번째 호출 형태: 종합시세 창에 있는 심볼 중 하나
   datetime    time2=TimeCurrent(tm);  // 두 번째 호출 형식: MqlDateTime 구조를 채운 Mark

//--- 날짜/시간을 수신하고 해당 데이터로 구조를 로그에 채운 결과를 표시
   PrintFormat("Tick time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\n-
               (string)time1, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day
  /*
  결과:
   Tick time: 2024.04.18 15:40:06
   - Year: 2024
   - Month: 04
   - Day: 18
```

© 2000-2025, MetaQuotes Ltd.

```
   - Hour: 15
   - Min: 40
   - Sec: 06
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeTradeServer

거래 서버의 계산된 현재 시간을 반환. TimeCurrent()와 달리 시간 값 계산은 클라이언트 터미널에서 수행되며 컴퓨터의 시간 설정에 따라 달라집니다. 함수의 2가지 변수가 있습니다.

매개변수 없이 호출

```
datetime  TimeTradeServer();
```

MqlDateTime 유형 매개변수를 사용하여 호출

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // 구조 유형의 변수
   );
```

매개변수

dt_struct

[out] 구조 유형의 변수 MqlDateTime.

값 반환

datetime 유형의 값

주의

MqlDateTime 구조 유형 변수가 매개변수로 전달된 경우 그에 따라 채워집니다.

고해상도 카운터 및 타이머를 정렬하려면 밀리초 단위의 값을 생성하는 GetTickCount() 함수를 사용합니다.

전략 테스터에서 테스트하는 동안 TimeTradeServer() 는 내역 데이터에 따라 시뮬레이션 되며 항상 TimeCurrent()와 동일합니.

예:

```
void OnStart()
  {
//--- 날짜/시간 데이터로 채워질 MqlDateTime 변수를 선언하고 마지막 쿼트 시간과 거래 서버의 예상
   MqlDateTime tm={};
   datetime    time_current=TimeCurrent();                  // 첫 번째 호출 형식: 종합시
   datetime    time_server =TimeTradeServer(tm);            // 두 번째 호출 형식: MqlDat
   int         difference  =int(time_current-time_server);  // TimeCurrent와 TimeTra

//--- 마지막 쿼트 시간과 거래 서버의 현재 예상 시간을 로그에 채워진 MqlDateTime 구조의 데이터로
   PrintFormat("Time Current: %s\nTime Trade Server: %s\n- Year: %u\n- Month: %02u\n-
               "- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of
               (string)time_current, (string)time_server, tm.year, tm.mon, tm.day, tm.
               EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week), difference);
  /*
  결과:
   Time Current: 2024.04.18 16:10:14
   Time Trade Server: 2024.04.18 16:10:15
   - Year: 2024
```

© 2000-2025, MetaQuotes Ltd.

```
   - Month: 04
   - Day: 18
   - Hour: 16
   - Min: 10
   - Sec: 15
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   Difference between Time Current and Time Trade Server: -1
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeLocal

클라이언트 터미널이 실행 중인 시스템의 로컬 시간을 반환. 함수의 2가지 변수가 있습니다.

매개변수 없이 호출

```
datetime  TimeLocal();
```

MqlDateTime 유형 매개변수를 사용하여 호출

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // 구조 유형의 변수
   );
```

매개변수

dt_struct

[out] 구조 유형의 변수 MqlDateTime.

값 반환

datetime 유형의 값

주의

MqlDateTime 구조 유형 변수가 매개변수로 전달된 경우 그에 따라 채워집니다.

고해상도 카운터 및 타이머를 정렬하려면 밀리초 단위의 값을 생성하는 GetTickCount() 함수를 사용합니다.

전략 테스터에서 테스트하는 동안 TimeLocal()은 항상 TimeCurrent() 시뮬레이션된 서버 시간과 같습니다.

예:

```
void OnStart()
  {
//--- PC 로컬 시간 데이터로 채워질 MqlDateTime 변수를 선언합니다.
   MqlDateTime tm={};
   datetime    time1=TimeLocal();   // 첫 번째 호출 형식: PC 로컬 시간
   datetime    time2=TimeLocal(tm); // 두 번째 호출 형식: MqlDateTime 구조를 채우는 PC 로

//--- PC 로컬 시간을 수신하고 로그의 해당 데이터로 구조를 채운 결과를 표시합니다.
   PrintFormat("Local time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\r
               (string)time1, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day
  /*
   결과:
    Local time: 2024.04.18 19:44:09
    - Year: 2024
    - Month: 04
    - Day: 18
    - Hour: 19
    - Min: 44
    - Sec: 09
```

© 2000-2025, MetaQuotes Ltd.

```
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeGMT

클라이언트 터미널이 실행 중인 시스템의 로컬 시간으로 DST 스위치를 고려하여 계산된 GMT를 반환. 함수의 2가지 변수가 있습니다.

매개변수 없이 호출

```
datetime  TimeGMT();
```

MqlDateTime 유형 매개변수를 사용하여 호출

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // 구조 유형의 변수
   );
```

매개변수

dt_struct

[out] 구조 유형의 변수 MqlDateTime.

값 반환

datetime 유형의 값

주의

MqlDateTime 구조 유형 변수가 매개변수로 전달된 경우 그에 따라 채워집니다.

고해상도 카운터 및 타이머를 정렬하려면 밀리초 단위의 값을 생성하는 GetTickCount() 함수를 사용합니다.

전략 테스터에서 테스트하는 동안 TimeGMT() 는 항상 TimeTradeServer() 시뮬레이션된 서버 시간과 같습니다.

예:

```
void OnStart()
  {
//--- 날짜/시간 데이터로 채워질 MqlDateTime 변수를 선언하고 PC 로컬 시간과 GMT를 가져옵니다.
   MqlDateTime tm={};
   datetime    time1=TimeLocal();            // 첫 번째 호출 형식: PC 로컬 시간
   datetime    time2=TimeGMT(tm);            // 두 번째 호출 형식: MqlDateTime 구조를 채우
   int         shift=int(time1-time2)/3600;  // GMT를 기준으로 한 로컬 시간 오프셋

//--- 로그에 채워진 MqlDateTime 구조의 데이터로 로컬 시간과 GMT를 표시합니다.
   PrintFormat("Time Local: %s\nTime GMT: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n
               "- Hour: %02u\n- Min: %02u\n- Sec: %02u\n- Day of Year: %03u\n- Day of
               (string)time1, (string)time2, tm.year, tm.mon, tm.day, tm.hour, tm.min,
               EnumToString((ENUM_DAY_OF_WEEK)tm.day_of_week), shift);
  /*
  결과:
   Time Local: 2024.04.18 19:37:23
   Time GMT: 2024.04.18 12:37:23
   - Year: 2024
```

© 2000-2025, MetaQuotes Ltd.

```
   - Month: 04
   - Day: 18
   - Hour: 12
   - Min: 37
   - Sec: 23
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
   Local time offset relative to GMT: +7
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeDaylightSavings

서머타임으로 전환한 경우 데이라이트 세이빙 시간(초)에 대한 보정을 반환. 컴퓨터의 시간 설정에 따라 다릅니다.

```
int  TimeDaylightSavings();
```

값 반환

윈터타임(표준)으로 전환한 경우 0을 반환.

예:

```
void OnStart()
  {
//--- 서머 타임 조정을 초 단위로 가져옵니다.
   int sec_dl=TimeDaylightSavings();

//--- 수신된 값을 설명하는 텍스트를 생성합니다.
   string text=(sec_dl==0 ? "Standard \"winter\" time is used" :
                StringFormat("Daylight saving time has been switched over. The correct

//--- 로그에 서머 타임 조정에 대한 설명을 초 단위로 표시합니다.
   Print(text);
   /*
   Result for "winter" time:
   Standard "winter" time is used

   Result for "summer" time:
   Daylight saving time has been switched over. The correction is -3600 seconds
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeGMTOffset

윈터타임 또는 서머타임으로 전환을 고려하여 GMT 시간과 로컬 컴퓨터 시간 간의 현재 차이를 초 단위로 반환. 컴퓨터의 시간 설정에 따라 다릅니다.

```
int  TimeGMTOffset();
```

값 반환

GMT 시간과 시스템 TimeLocal의 현재 시간 간의 차이(초)를 나타내는 int 유형의 값입니다.

```
TimeGMTOffset() =  TimeGMT() - TimeLocal()
```

예:

```
void OnStart()
  {
//--- 로컬 시간, GMT 및 GMT와 로컬 컴퓨터 시간의 차이를 초 단위로 가져옵니다.
   datetime time_local=TimeLocal();
   datetime time_gmt  =TimeGMT();
   int      offset    =TimeGMTOffset();

//--- 얻은 시간 및 오프셋 값을 로그에 표시합니다.
   PrintFormat("Local Time: %s, GMT Time: %s, Seconds Offset: %+d", (string)time_local
  /*
  결과:
   Local Time: 2024.04.18 19:35:52, GMT Time: 2024.04.18 12:35:52, Seconds Offset: -25
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

TimeToStruct

날짜 시간 유형의 값(1970년 1월 1일 이후의 초 단위 시간)을 구조 변수 MqlDateTime으로 반환.

```
bool  TimeToStruct(
   datetime      dt,            // 날짜 및 시간
   MqlDateTime&  dt_struct      // 가치의 채택을 위한 구조
   );
```

매개변수

dt

[in] 변환할 날짜 값.

dt_struct

[out] 구조 유형 MqlDateTime의 변수.

값 반환

성공하면 true, 실패하면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

예:

```
void OnStart()
  {
//--- 서버의 마지막으로 알려진 시간을 가져오고 날짜/시간 구조를 선언하고 구조 필드를 채웁니다.
   datetime    time=TimeCurrent();
   MqlDateTime tm  ={};
   if(!TimeToStruct(time,tm))
      Print("TimeToStruct() failed. Error ", GetLastError());

//--- 획득한 서버 시간과 TimeToStruct()를 사용하여 MqlDateTime 구조를 채운 결과를 로그에 표시
   PrintFormat("Server time: %s\n- Year: %u\n- Month: %02u\n- Day: %02u\n- Hour: %02u\
               (string)time, tm.year, tm.mon, tm.day, tm.hour, tm.min, tm.sec, tm.day_
  /*
  결과:
   Server time: 2024.04.18 15:47:27
   - Year: 2024
   - Month: 04
   - Day: 18
   - Hour: 15
   - Min: 47
   - Sec: 27
   - Day of Year: 108
   - Day of Week: 4 (THURSDAY)
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

StructToTime

구조 변수 MqlDateTime을 datetime 유형의 값으로 변환하고 결과 값을 반환합니다.

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // 날짜 및 시간의 구조
   );
```

매개변수

dt_struct

[in] 구조 유형의 변수 MqlDateTime.

값 반환

1970년 1월 1일 이후 시간(초)을 포함하는 datetime 유형의 값.

예:

```
#property script_show_inputs

input int   InpYear  =  0;    연도
input int   InpMonth =  0;    // 월
input int   InpDay   =  0;    // 일
input int   InpHour  =  0;    // 시
input int   InpMin   =  0;    // 분
input int   InpSec   =  0;    // 초

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 입력된 값을 조정하고 변수에 씁니다.
   int year =  (InpYear<1970 ? 1970 : InpYear); // 입력한 연도가 1970년보다 작으면 1970년
   int month=  (InpMonth<1 ?  1  :  InpMonth > 12  ?  12 :  InpMonth);
   int day  =  (InpDay  <1 ?  1  :  InpDay   > 31  ?  31 :  InpDay);
   int hour =  (InpHour <0 ?  0  :  InpHour  > 23  ?  23 :  InpHour);
   int min  =  (InpMin  <0 ?  0  :  InpMin   > 59  ?  59 :  InpMin);
   int sec  =  (InpSec  <0 ?  0  :  InpSec   > 59  ?  59 :  InpSec);

//--- 입력된 값을 로그에 표시합니다.
   PrintFormat("Entered date and time: %04u.%02u.%02u %02u:%02u:%02u", InpYear, InpMor

//--- 조정된 입력 값을 로그에 표시합니다.
   PrintFormat("Corrected date and time: %04u.%02u.%02u %02u:%02u:%02u", year, month,

//--- 해당 구조 필드에 입력 값을 씁니다.
   MqlDateTime time_struct={};
   time_struct.year= year;
   time_struct.mon = month;
```

© 2000-2025, MetaQuotes Ltd.

```
   time_struct.day = day;
   time_struct.hour= hour;
   time_struct.min = min;
   time_struct.sec = sec;

//--- 구조체의 날짜와 시간을 datetime 유형의 변수로 변환하고
   datetime time = StructToTime(time_struct);

//--- MqlDateTime 구조체 유형 변수를 datetime 유형 값으로 변환한 결과를 표시합니다.
   Print("Converted date and time: ",TimeToString(time,TIME_DATE|TIME_MINUTES|TIME_SEC
  /*
  기본값이 0으로 입력되면 결과는 다음과 같습니다.
   Entered date and time: 0000.00.00 00:00:00
   Corrected date and time: 1970.01.01 00:00:00
   Converted date and time: 1970.01.01 00:00:00

  올해 2월의 날짜를 잘못 입력한 경우 결과:
   Entered date and time: 2024.02.31 00:00:00
   Corrected date and time: 2024.02.31 00:00:00
   Converted date and time: 2024.03.02 00:00:00
  */
  }
```

© 2000-2025, MetaQuotes Ltd.

계정 정보

현재 계정의 변수를 반환하는 기능.

기능	액션
AccountInfoDouble	해당 계정 속성의 double 유형 값을 반환
AccountInfoInteger	해당 계정 속성의 정수 유형(bool, int 또는 long) 값을 반환
AccountInfoString	해당 계정 속성의 string 유형 값을 반환

© 2000-2025, MetaQuotes Ltd.

AccountInfoDouble

해당 계정 속성의 값을 반환.

```
double  AccountInfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  property_id      // 속성 식별자
   );
```

매개변수

property_id

[in] 속성 식별자. 값은 ENUM_ACCOUNT_INFO_DOUBLE 값 중 하나일 수 있습니다.

값 반환

double 유형의 값.

예를 들어:

```
void OnStart()
  {
//--- AccountInfoDouble() 기능에서 사용할 수 있는 모든 정보 표시
   printf("ACCOUNT_BALANCE =  %G",AccountInfoDouble(ACCOUNT_BALANCE));
   printf("ACCOUNT_CREDIT =  %G",AccountInfoDouble(ACCOUNT_CREDIT));
   printf("ACCOUNT_PROFIT =  %G",AccountInfoDouble(ACCOUNT_PROFIT));
   printf("ACCOUNT_EQUITY =  %G",AccountInfoDouble(ACCOUNT_EQUITY));
   printf("ACCOUNT_MARGIN =  %G",AccountInfoDouble(ACCOUNT_MARGIN));
   printf("ACCOUNT_MARGIN_FREE =  %G",AccountInfoDouble(ACCOUNT_MARGIN_FREE));
   printf("ACCOUNT_MARGIN_LEVEL =  %G",AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));
   printf("ACCOUNT_MARGIN_SO_CALL = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));
   printf("ACCOUNT_MARGIN_SO_SO = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));
  }
```

추가 참조

SymbolInfoDouble, SymbolInfoString, SymbolInfoInteger, PrintFormat

© 2000-2025, MetaQuotes Ltd.

AccountInfoInteger

해당 계정 속성의 값을 반환.

```
long  AccountInfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  property_id      // 속성의 식별자
   );
```

매개변수

property_id

[in] 속성 식별자. 값은 ENUM_ACCOUNT_INFO_INTEGER 값 중 하나일 수 있습니다.

값 반환

long 유형의 값.

주의

속성은 반드시 bool, int 또는 long 유형이어야 합니다.

예를 들어:

```
void OnStart()
  {
//--- AccountInfoInteger() 기능에서 사용할 수 있는 모든 정보 표시
   printf("ACCOUNT_LOGIN =  %d",AccountInfoInteger(ACCOUNT_LOGIN));
   printf("ACCOUNT_LEVERAGE =  %d",AccountInfoInteger(ACCOUNT_LEVERAGE));
   bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
   bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
   ENUM_ACCOUNT_TRADE_MODE tradeMode=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(ACCOU
   ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteger

//--- 거래 작업을 수행할 수 있는 가능성에 대한 정보 제공
   if(thisAccountTradeAllowed)
      print("이 계정에 대한 거래가 허용됨");
   else
      Print("이 계정에 대한 거래는 금지됨!");

//--- 엑스퍼트 어드바이저가 이 계정을 거래할 수 있는지 확인
   if(EATradeAllowed)
      Print("이 계정에는 엑스퍼트 어드바이저에 의한 거래가 허용됨");
   else
      Print("이 계정에는 엑스퍼트 어드바이저 거래가 금지됨");

//--- 계정 유형 확인
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("데모 계정임");
         break;
      case(ACCOUNT_TRADE_MODE_CONTEST):
```

© 2000-2025, MetaQuotes Ltd.

```
         Print("경쟁 계정입니다");
         break;
      default:Print("실제 계정입니다!");
     }

//--- StopOut 레벨 설정 모드 확인
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("StopOut 레벨은 지정된 백분율입니다");
         break;
      default:Print("StopOut 레벨은 통화 용어로 지정됩니다");
     }
  }
```

추가 참조

계정 정보

© 2000-2025, MetaQuotes Ltd.

AccountInfoString

해당 계정 속성의 값을 반환.

```
string  AccountInfoString(
   ENUM_ACCOUNT_INFO_STRING  property_id      // 속성 식별자
   );
```

매개변수

property_id

[in] 속성 식별자. 값은 ENUM_ACCOUNT_INFO_STRING의 값 중 하나일 수 있습니다.

값 반환

string 유형의 값.

예를 들어:

```
void OnStart()
  {
//--- AccountInfoString() 함수에서 사용할 수 있는 모든 정보 표시
   Print("브로커 이름 = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("예금 통화 = ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("클라이언트 이름 = ",AccountInfoString(ACCOUNT_NAME));
   Print("거래 서버의 이름 = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

추가 참조

계정 정보

© 2000-2025, MetaQuotes Ltd.

상태 점검

클라이언트 터미널의 현재 상태 변수를 반환하는 기능

기능	액션
GetLastError	마지막 오류 반환
IsStopped	mql5 프로그램이 작동을 중지하도록 명령을 받은 경우 참값을 반환
UninitializeReason	초기화 취소 사유 코드를 반환
TerminalInfoInteger	mql5 프로그램 환경의 해당 속성의 정수 값을 반환
TerminalInfoDouble	mql5 프로그램 환경의 해당 속성의 이중 값을 반환
TerminalInfoString	mql5 프로그램 환경의 해당 속성의 string 값을 반환
MQLInfoInteger	실행 중인 mql5 프로그램의 해당 속성의 정수 값을 반환
MQLInfoString	실행 중인 mql5 프로그램의 해당 속성의 문자열 값을 반환
심볼	현재 차트의 심볼 이름을 반환
Period	현재 차트 타임프레임을 반환
Digits	현재 차트 심볼 가격 값의 정확도를 결정하는 소수 자릿수를 반환
Point	시세 통화에서 현재 심볼의 포인트 크기를 반환

© 2000-2025, MetaQuotes Ltd.

GetLastError

시스템 변수 _LastError의 내용을 반환.

```
int  GetLastError();
```

값 반환

mql5 프로그램을 실행하는 동안 발생한 마지막 오류 값을 반환.

주의

함수 호출 후에는 _LastError의 내용이 재설정되지 않습니다. 이 변수를 재설정하려면 ResetLastError() 호출해야 합니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[1]={};   // 현재 바 데이터를 여기에 표시

//--- 일부러 부적절한 매개변수를 사용하여 함수를 호출합니다.
   int res=CopyRates(NULL, PERIOD_CURRENT, 0, 2, rates);
   if(res!=2)
      PrintFormat("CopyRates() returned %d. LastError %d", res, GetLastError());

//--- 현재 바 데이터를 MqlRates 구조에 복사하기 전에 마지막 오류 코드를 재설정합니다.
   ResetLastError();
//--- 함수가 제대로 작동하지 않으면 오류 코드는 0이 아닙니다.
   CopyRates(NULL, PERIOD_CURRENT, 0, 1, rates);
   Print("CopyRates() error ", GetLastError());

//--- 얻은 값의 배열을 인쇄합니다.
   ArrayPrint(rates);
  }
```

추가 참조

거래 서버 반환 코드

© 2000-2025, MetaQuotes Ltd.

IsStopped

mql5 프로그램의 강제 종료 여부를 확인.

```
bool  IsStopped();
```

값 반환

만약 _StopFlag 시스템 변수에 0이 아닌 값이 포함되어 있으면 true를 반환. mql5 프로그램이 작업을 완료하도록 명령을 받은 경우, 0이 아닌 값이 _StopFlag에 기록됩니다. 이 경우 프로그램을 즉시 종료해야 하며, 그렇지 않으면 프로그램이 3초 후에 외부에서 강제로 완료됩니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 스탑 확인이 포함된 무한 루프
   while(!IsStopped())
     {
//--- 차트에 로컬 PC 시간을 표시합니다.
      Comment("Time Local: ", TimeToString(TimeLocal(), TIME_DATE|TIME_MINUTES|TIME_SE
      Sleep(16);
     }
   Print("The StopFlag is set. 프로그램이 중단됩니다.");

//--- 모두 삭제
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

UninitializeReason

초기화 취소 사유코드를 반환.

```
int  UninitializeReason();
```

값 반환

형성된_UninitReason 값을 OnDeinit()가 호출되기 전에 반환. 값은 초기화를 취소한 사유에 따라 달라집니다.

예:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 초기화 해제 이유 코드를 'reason' 변수로 가져옵니다.
   int reason=UninitializeReason();
//--- 초기화 해제 이유가 포함된 메시지 문자열을 생성하고 해당 메시지를 저널에 보냅니다.
   string message=StringFormat("%s: Uninitialize reason code: %d (%s)",__FUNCTION__, r
   Print(message);
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 'reason' 형식 변수에서 초기화 해제 이유 코드를 포함하는 메시지 문자열을 생성하고 해당 메시
   string message=StringFormat("%s: Uninitialize reason code: %d (%s)",__FUNCTION__, r
   Print(message);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
  }
//+------------------------------------------------------------------+
//| 초기화 해제 이유 코드에 대한 설명을 반환합니다.                                      |
//+------------------------------------------------------------------+
string UninitializeReasonDescription(const int reason)
  {
   switch(reason)
     {
 //--- EA가 ExpertRemove() 함수 호출 작업을 중지했습니다
      case REASON_PROGRAM :
```

© 2000-2025, MetaQuotes Ltd.

```
 return("Expert Advisor가 ExpertRemove() 함수를 호출하여 작업을 종료했습니다.");< /t4>
      //--- 차트에서 삭제된 프로그램
      case REASON_REMOVE :
 return("프로그램이 차트에서 삭제되었습니다.");
      //--- 다시 컴파일 된 프로그램
      case REASON_RECOMPILE :
        return("Program has been recompiled");
 //--- 심볼 또는 차트 기간이 변경됨
      case REASON_CHARTCHANGE :
        return("Symbol or chart period has been changed");
      //--- 차트 종료됨
      case REASON_CHARTCLOSE :
        return("Chart has been closed");
      //--- 유저에 의해 변경된 입력
      case REASON_PARAMETERS :
        return("Input parameters have been changed by a user");
 //--- 다른 계정이 활성화되었거나 계정 설정 변경으로 인해 거래 서버에 다시 연결되었습니다
      case REASON_ACCOUNT :
        return("Another account has been activated or reconnection to the trade server
      //--- 다른 차트 탬플릿이 적용됨
      case REASON_TEMPLATE :
        return("A new template has been applied");
 //--- OnInit() 핸들러가 0이 아닌 값을 반환했습니다.
      case REASON_INITFAILED :
        return("This value means that OnInit() handler has returned a nonzero value");
      //--- 터미널이 종료되었습니다
      case REASON_CLOSE :
        return("Terminal has been closed");
     }

//--- 초기화 해제 이유를 알 수 없음
   return("Unknown reason");
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoInteger

mql5 프로그램 환경의 해당 속성 값을 반환.

```
int  TerminalInfoInteger(
   int  property_id      // 속성의 식별자
   );
```

매개변수

property_id

[in] 속성의 식별자. Can be one of the values of the ENUM_TERMINAL_INFO_INTEGER 열거 값 중 하나일 수 있습니다.

값 반환

int 유형의 값.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 실행 중인 터미널의 빌드 번호와 "64비트 터미널" 속성을 가져옵니다.
   int  build = TerminalInfoInteger(TERMINAL_BUILD);
   bool x64   = TerminalInfoInteger(TERMINAL_X64);

//--- 획득한 터미널 데이터를 저널에 인쇄합니다.
   PrintFormat("MetaTrader 5 %s build %d", (x64 ? "x64" : "x32"), build);
   /*
  결과:
   MetaTrader 5 x64 build 4330
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoDouble

mql5 프로그램 환경의 해당 속성 값을 반환.

```
double  TerminalInfoDouble(
   int  property_id      // 속성의 식별자
   );
```

매개변수

property_id

[in] 속성의 식별자. ENUM_TERMINAL_INFO_DOUBLE 열거 값 중 하나일 수 있습니다.

값 반환

double 유형의 값.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 실행 중인 터미널의 빌드 번호와 "64비트 터미널" 속성을 가져옵니다.
   int  build = TerminalInfoInteger(TERMINAL_BUILD);
   bool x64   = TerminalInfoInteger(TERMINAL_X64);

//--- 획득한 터미널 데이터를 저널에 인쇄합니다.
   PrintFormat("MetaTrader 5 %s build %d", (x64 ? "x64" : "x32"), build);
   /*
  결과:
   MetaTrader 5 x64 build 4330
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

TerminalInfoString

mql5 프로그램 환경의 해당 속성 값을 반환. 속성은 문자열 유형이어야 합니다.

```
string  TerminalInfoString(
   int  property_id      // 속성의 식별자
   );
```

매개변수

property_id

[in] 속성의 식별자. ENUM_TERMINAL_INFO_STRING 열거 값 중 하나일 수 있습니다.

값 반환

string 유형의 값.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 운영체계와 터미널 데이터를 가져옵니다
   string os_ver  = TerminalInfoString(TERMINAL_OS_VERSION);       // 유저의 운영 체계
   string name    = TerminalInfoString(TERMINAL_NAME);             // 터미널 이름
   string path    = TerminalInfoString(TERMINAL_PATH);             // 터미널이 실행된 폴더
   string data    = TerminalInfoString(TERMINAL_DATA_PATH);        // 터미널 데이터를 저장
   string common  = TerminalInfoString(TERMINAL_COMMONDATA_PATH); // 컴퓨터에 설치된 모든

//--- 얻은 데이터를 저널로 보냅니다.
   PrintFormat("OS: %s\nTerminal: %s\n- Path: %s\n- Data path: %s\n- Common Data path:
   /*
   OS: Windows 10 build 19045
   Terminal: MetaTrader 5
   - Path: E:\MetaQuotes\MetaTrader 5
   - Data path: E:\MetaQuotes\MetaTrader 5
   - Common Data path: C:\Users\admin\AppData\Roaming\MetaQuotes\Terminal\Common
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

MQLInfoInteger

실행 중인 mql5 프로그램의 해당 속성 값을 반환.

```
int  MQLInfoInteger(
   int  property_id      // 속성의 식별자
   );
```

매개변수

property_id

[in] 속성의 식별자. ENUM_MQL_INFO_INTEGER 열거의 값 중 하나일 수 있습니다.

값 반환

int 유형의 값.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- MQL 프로그램에 사용 가능한 메모리 값과 소비되는 메모리 값을 가져옵니다.
   int limit = MQLInfoInteger(MQL_MEMORY_LIMIT);   // MQL5 프로그램에 대해 가능한 최대 동
   int used  = MQLInfoInteger(MQL_MEMORY_USED);    // MQL5 프로그램에서 사용하는 메모리 크

//--- 수신된 값을 저널로 보냅니다.
   PrintFormat("Maximum possible amount of dynamic memory for MQL5 program: %d Mb\n"+
               "Memory used by MQL5 program: %d Mb", limit, used);
   /*
  결과
   Maximum possible amount of dynamic memory for MQL5 program: 8388608 Mb
   Memory used by MQL5 program: 2 Mb
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

MQLInfoString

실행 중인 mql5 프로그램의 해당 속성 값을 반환.

```
string  MQLInfoString(
   int  property_id      // 속성의 식별자
   );
```

매개변수

property_id

[in] 속성의 식별자. ENUM_MQL_INFO_STRING 열거의 하나일 수 있습니다.

값 반환

string 유형의 값.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 시작된 프로그램 데이터를 가져옵니다.
   string name = MQLInfoString(MQL_PROGRAM_NAME);  // 시작된 MQL5 프로그램의 이름
   string path = MQLInfoString(MQL_PROGRAM_PATH);  // 실행중인 프로그램의 경로

//--- 얻은 데이터를 저널로 보냅니다.
   PrintFormat("Name of the running MQL program: '%s'\nPath of the running MQL progra
   /*
  결과:
   Name of the running MQL program: 'MQLInfoString'
   Path of the running MQL program: E:\MetaQuotes\MetaTrader 5\MQL5\Scripts\MQLInfoSt
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

심볼

현재 차트의 심볼 이름을 반환.

```
string  Symbol();
```

값 반환

현재 차트 심볼 이름을 저장하는 _Symbol 시스템 변수 값.

주의

엑스퍼트 어드바이저, 지표 및 스크립트와는 달리 서비스는 특정 차트에 바인딩되지 않습니다. 그러므로, Symbol() 는 서비스에 대해 빈 문자열 ("")을 반환합니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 기호 이름을 가져옵니다.
   string name = Symbol();

//--- 얻은 데이터를 저널로 보냅니다.
   PrintFormat("Current chart symbol name: '%s'", name);
   /*
  결과
   Current chart symbol name: 'EURUSD'
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Period

현재 차트 시간프레임을 반환.

```
ENUM_TIMEFRAMES  기간();
```

값 반환

현재 차트 시간프레임을 포함하는 _Period 변수의 내용. ENUM_TIMEFRAMES 열거 값 중 하나의 값일 수 있습니다.

주의

엑스퍼트 어드바이저, 지표 및 스크립트와는 달리 서비스는 특정 차트에 바인딩되지 않습니다. 그러므로, Period() 은 서비스에 대해 0을 반환합니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트의 기간 값과 설명을 가져옵니다.
   ENUM_TIMEFRAMES period    = Period();
   string          timeframe = StringSubstr(EnumToString(period), 7);

//--- 얻은 데이터를 저널로 보냅니다.
   PrintFormat("Current chart timeframe: %s\nTimeframe value: %s (%d)",
               timeframe, EnumToString(period), period);
   /*
  결과:
   Current chart timeframe: H4
   Timeframe value: PERIOD_H4 (16388)
   */
  }
```

추가 참조

PeriodSeconds, 차트 시간프레임, 날짜 및 시간, 객체의 가시성

© 2000-2025, MetaQuotes Ltd.

Digits

현재 차트 심볼의 가격 정확도를 결정하는 소수 자릿수를 반환.

```
int  Digits();
```

값 반환

현재 차트 심볼의 가격 정확도를 결정하는 소수 자릿수를 저장하는_Digits변수의 값.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 심볼의 소수점 이하 자릿수를 가져옵니다.
   int digits = Digits();

//--- 얻은 데이터를 저널로 보냅니다.
   Print("Number of decimal digits for the current chart symbol: ", digits);
   /*
  결과:
   Number of decimal digits for the current chart symbol: 5
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

Point

시세 통화에서 현재 심볼에 대한 포인트 크기를 반환.

```
double  Point();
```

값 반환

현재 심볼의 포인트 크기를 시세 통화에 저장하는 _Point 변수의 값 입니다.

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 통화에서 현재 심볼 포인트 크기를 가져옵니다.
   double point = Point();

//--- 얻은 데이터를 저널로 보냅니다.
   Print("Point size of the current symbol in the quote currency: ", DoubleToString(po
   /*
  결과:
   Point size of the current symbol in the quote currency: 0.00001
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

이벤트 핸들링

MQL5 언어는 특정 사전 정의된 이벤트를 처리합니다. 이러한 이벤트를 처리하는 기능은 MQL5 프로그램에 정의되어야 합니다. 함수 이름, 반환 유형, 매개변수 세트(있는 경우)가 있으며, 해당 유형은 이벤트 처리 기능의 설명과 엄격히 일치해야 합니다.

클라이언트 터미널 이벤트 처리기는 반환 및 매개변수 유형을 사용하여 이벤트를 처리하는 기능을 식별합니다. 특정 함수에 일부 매개변수가 있거나 아래 설명과 일치하지 않는 반환 유형이 있는 경우 이러한 기능을 사용하여 이벤트를 처리할 수 없습니다.

기능	설명
OnStart	이 기능은 스크립트에 설정된 작업을 수행하기 위해 시작 이벤트가 발생하면 호출됩니다
OnInit	이 기능은 MQL5 프로그램을 초기화하기 위해 Init 이벤트가 발생하면 지표 및 EA에 호출됩니다
OnDeinit	이 기능은 시작된 MQL5 프로그램을 초기화 하기 위해 Deinit 이벤트가 발생하면 지표 및 EA에 호출됩니다
OnTick	이 기능은 새 시세를 핸들링하기 위해 NewTick 이벤트가 발생하면 EA에 호출됩니다
OnCalculate	이 기능은 가격 데이터 변경을 처리하기 위해 계산 이벤트가 발생하면 지표에 호출됩니다
OnTimer	이 기능은 고정된 시간 간격으로 터미널에 의해 생성된 타이머 주기 이벤트동안 지표와 EA에 호출됩니다
OnTrade	이 기능은 거래 서버에서 트레이딩 작업이 끝날 때 생성된 트레이드 이벤트 중에 EA에 호출됩니다
OnTradeTransaction	이 기능은 거래 요청 실행 결과를 처리하기 위해 거래 트랜잭션 이벤트가 발생할 때 EA에 호출됩니다
OnBookEvent	이 기능은 시장 깊이 변화를 처리하기 위해 BookEvent 이벤트가 발생하면 EA에 호출됩니다
OnChartEvent	이 기능은 사용자 또는 MQL5 프로그램에 의해 수행된 차트 변경을 처리 하기 위해 ChartEvent 이벤트가 발생하면 지표 및 EA에 호출됩니다
OnTester	이 기능은 내역 데이터에 대해 EA를 테스트를 한 후 테스터 이벤트가 발생하면 EA에 호출됩니다
OnTesterInit	이 기능은 전략 테스터에서 최적화하기 전에 필요한 작업을 수행하기 위해 TesterInit 이벤트가 발생하면 EA에 호출됩니다
OnTesterDeinit	이 기능은 전략 테스터에서 EA 최적화 후 TesterDeinit 이벤트가 발생하면 EA에 호출됩니다
OnTesterPass	이 기능은 전략 테스터에서 EA 최적화 중 새 데이터 프레임 도착을 처리하기 위해 TesterPass 이벤트가 발생하면 EA에 호출됩니다

© 2000-2025, MetaQuotes Ltd.

클라이언트 터미널이 열린 해당 차트에 수신 이벤트를 보냅니다. 또한 차트 (차트 이벤트) 또는 mql5 프로그램 (사용자 지정 이벤트)에 의해 이벤트가 생성될 수 있습니다. 그래픽 개체 생성/삭제 이벤트 생성은 CHART_EVENT_OBJECT_CREATE 및 CHART_EVENT_OBJECT_DELETE 차트 속성을 설정하여 활성화/비활성화할 수 있습니다. 각 mql5 애플리케이션 및 차트에는 새로 도착한 모든 이벤트가 배치되는 자체 이벤트 대기열이 있습니다.

프로그램은 실행 중인 차트에서만 이벤트를 가져옵니다. 모든 이벤트는 영수증 순서에 따라 하나씩 처리됩니다. 대기열에 이미 NewTick 이벤트가 포함되어 있거나 이 이벤트가 처리 중인 경우 새 NewTick 이벤트는 mql5애플리케이션 대기열에 추가되지 않습니다. 마찬가지로 ChartEvent가 이미 mql5 프로그램 대기열에 있거나 이러한 이벤트를 처리하고 있는 경우 이 유형의 새 이벤트가 대기열에 배치되지 않습니다. 타이버 이벤트 처리는 동일한 방식으로 처리되며 타이머 이벤트가 이미 대기열에 있거나 처리 중인 경우 새 타이머 이벤트가 대기열에 설정되지 않습니다.

이벤트 대기열은 크기가 제한적이지만 충분하므로 올바르게 개발된 프로그램에서 대기열 오버플로우가 발생할 가능성이 낮습니다. 대기열이 오버플로우 되면 새 이벤트가 대기열에 설정되지 않고 삭제됩니다.

무한 루프를 사용하여 이벤트를 처리하지 않는 것이 좋습니다. 단일 시작 이벤트를 처리하는 스크립트는 예외일 수 있습니다.

라이브러리는 이벤트를 처리하지 않습니다.

© 2000-2025, MetaQuotes Ltd.

OnStart

시작 이벤트가 발생하면 스크립트와 서비스에서 이 함수가 호출됩니다. 이 함수는 프로그램에 구현된 작업의 일회성 실행을 위한 것입니다. 두 가지 함수 유형이 있습니다.

결과를 반환하는 버전

```
int  OnStart(void);
```

반환 값

int 타입의 값이 Journal 탭에 표시됩니다.

스크립트 실행이 완료된 후 터미널 저널에 "script script_name removed (result code N)" 항목이 생성됩니다. 여기서 N은 OnStart() 함수로 반환되는 값입니다.

서비스 실행 완료 후 터미널 저널에 "service service_name stopped (result code N)" 항목이 생성됩니다. 여기서 N은 OnStart() 함수로 반환되는 값입니다.

실행 결과를 반환하는 OnStart() 호출은 스크립트 또는 서비스 실행을 허용할 뿐만 아니라 프로그램 실행 결과를 분석하는 데 유용한 오류 코드나 기타 데이터를 반환하므로 사용하는 것이 좋습니다.

결과 반환 없는 버전은 이전 코드와의 호환성을 위해서만 유지됩니다. 사용을 권장하지 않습니다

```
void  OnStart(void);
```

참고

OnStart()는 스크립트와 서비스의 이벤트를 처리하는 유일한 함수입니다. 이 프로그램에는 다른 이벤트가 전송되지 않습니다. 또한 시작 이벤트는 EA 및 커스텀 지표로 전달되지 않습니다.

샘플 스크립트:

```
//--- 색상 작업을 위한 매크로
#define XRGB(r,g,b)     (0xFF000000|(uchar(r)<<16)|(uchar(g)<<8)|uchar(b))
#define GETRGB(clr)     ((clr)&0xFFFFFF)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 하향 캔들의 색상 설정
   Comment("하향 캔들의 색상 설정");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BEAR,GetRandomColor());
   ChartRedraw(); // 새 틱을 기다리지 마시고 즉시 차트를 업데이트하십시오
   Sleep(1000);   // 모든 변경 내용을 보려면 1초간 일시 중지하십시오
//--- 상향 캔들의 색상 설정
   Comment("상향 캔들의 색상 설정");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BULL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 배경 색상 설정
   Comment("배경 색상 설정");
   ChartSetInteger(0,CHART_COLOR_BACKGROUND,GetRandomColor());
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartRedraw();
   Sleep(1000);
//--- Ask line의 색상 설정
   Comment("Ask line의 색상 설정");
   ChartSetInteger(0,CHART_COLOR_ASK,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- Bid line의 색상 설정
   Comment("Bid line의 색상 설정");
   ChartSetInteger(0,CHART_COLOR_BID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 하향 막대와 하향 캔들 프레임의 색상 설정
   Comment("하향 막대와 하향 캔들 프레임의 색상 설정");
   ChartSetInteger(0,CHART_COLOR_CHART_DOWN,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 차트 라인과 Doji 캔들스틱의 색상 설정
   Comment("차트 라인과 Doji 캔들스틱의 색상 설정");
   ChartSetInteger(0,CHART_COLOR_CHART_LINE,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 상향 막대와 상향 캔들 프레임의 색상 설정
   Comment("상향 막대와 상향 캔들 프레임의 색상 설정");
   ChartSetInteger(0,CHART_COLOR_CHART_UP,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 축, 스케일 및 OHLC 라인의 색상 설정
   Comment("축, 스케일 및 OHLC 라인의 색상 설정");
   ChartSetInteger(0,CHART_COLOR_FOREGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 그리드 색상 설정
   Comment("그리드 색상 설정");
   ChartSetInteger(0,CHART_COLOR_GRID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 전날 종가 색상 설정
   Comment("전날 종가 색상 설정");
   ChartSetInteger(0,CHART_COLOR_LAST,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 스탑로스 및 이익실현 주문 수준 색상 설정
   Comment("스탑로스 및 이익실현 주문 수준 색상 설정");
   ChartSetInteger(0,CHART_COLOR_STOP_LEVEL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- 거래량 및 시장 진입 수준의 색상 설정
   Comment("거래량 및 시장 진입 수준의 색상 설정");
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartSetInteger(0,CHART_COLOR_VOLUME,GetRandomColor());
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| 임의로 생성된 색상 반환                                          |
//+------------------------------------------------------------------+
color GetRandomColor()
  {
   color clr=(color)GETRGB(XRGB(rand()%255,rand()%255,rand()%255));
   return clr;
  }
```

더 보기

Event handling functions, Program running, Client terminal events

© 2000-2025, MetaQuotes Ltd.

OnInit

이 함수는 Init 이벤트가 발생할 때 지표 및 EA로 호출됩니다. 실행 중인 MQL5 프로그램을 초기화하는 데 사용됩니다. 두 가지 함수 유형이 있습니다.

결과를 반환하는 버전

```
int  OnInit(void);
```

반환 값

int 타입 값, 0은 초기화가 성공했음을 의미합니다.

INIT_FAILED를 반환하면 EA가 차트에서 강제로 언로드됩니다.

INIT_FAILED를 반환하면 지표가 차트에서 언로드되지 않습니다. 차트에 남아 있는 지표는 작동하지 않습니다. —이벤트 핸들러는 지표에서 호출되지 않습니다.

실행 결과를 반환하는 OnInit() 호출은 프로그램 초기화를 허용할 뿐 아니라 프로그램이 조기에 종료되는 경우 오류 코드를 반환하므로 사용하는 것이 좋습니다.

결과 반환 없는 버전은 이전 코드와의 호환성을 위해서만 유지됩니다. 사용을 권장하지 않습니다

```
void  OnInit(void);
```

참고

Init 이벤트는 EA 또는 지표를 로드한 후 즉시 생성됩니다. 스크립트에 대한 이벤트가 생성되지 않았습니다. OnInit() 함수는 MQL5 프로그램을 초기화하는 데 사용됩니다. OnInit()의 반환 값이 int 유형인 경우, 0이 아닌 반환 코드는 초기화에 실패했음을 의미하며 Deinit 이벤트를 REASON_INITFAILED 초기화 해제 근거 코드를 사용하여 생성합니다.

OnInit()의void 타입의 함수는 항상 성공적인 초기화를 의미하며 사용하지 않는 것이 좋습니다.

최적화를 EA 입력에 대해서 하려면, 반환 코드로 ENUM_INIT_RETCODE 열거값의 값을 사용하는 것이 좋습니다. 이러한 값은 가장 적합한 테스트 에이전트의 선택을 포함하여 최적화 프로세스 관리를 확립하기 위한 것입니다. 테스트를 시작하기 전에 EA 초기화 중 TerminalInfoInteger() 함수를 사용하여 에이전트 구성 및 리소스(코어 수, 사용가능한 메모리 양 등)에 대한 데이터를 요청할 수 있습니다. 획득한 데이터에 따라 테스트 에이전트 사용을 허용하거나 EA 최적화를 금지할 수 있습니다.

ID	Description
INIT_SUCCEEDED	초기화에 성공하면 EA 테스트를 계속할 수 있습니다. 이 코드는 0 값과 동일함을 의미합니다. 테스터에서 EA가 성공적으로 초기화되었습니다.
INIT_FAILED	초기화 실패. 어쩔 수 없는 오류로 인해 테스트를 계속하는 것은 의미가 없다. 예를 들어 EA 작업에 필요한 지표를 생성할 수 없습니다. 이 값의 반환은 0과 다른 값을 반환하는 것과 동일함을 의미합니다. 테스터에서 EA 초기화가 실패했습니다.
INIT_PARAMETERS_INCORRECT	프로그래머의 잘못된 입력 매개변수 집합을 나타내도록 설계되었습니다. 일반 최적화 표에서 이 반환 코드가 포함된 결과 문자열이 빨간색으로 강조 표시됩니다.

© 2000-2025, MetaQuotes Ltd.

ID	Description
	그러한 EA 입력 세트에 대한 테스트는 수행되지 않습니다. 에이전트가 새 태스크를 수신할 준비가 되었습니다. 이 값이 수신되면 전략 테스터는 반복 실행을 위해 이 태스크를 다른 에이전트에 전달하지 않습니다.
INIT_AGENT_NOT_SUITABLE	초기화 중 프로그램 실행 오류가 없습니다. 그러나 어떤 이유로 이 에이전트는 테스트를 수행하기에 적합하지 않습니다. 예를 들어 RAM이 충분하지 않거나, OpenCL 지원하지 않음, 등이 있습니다. 이 코드를 반환한 후, 에이전트는 이 최적화가 끝날 때까지 더 이상 태스크를 수신하지 않습니다.

OnInit(), INIT_FAILED/INIT_PARAMETERS_INCORRECT을 테스터에서 반환하는 함수, 는 EA를 최적화할 때 고려해야 할 몇 가지 특성이 있습니다:

- OnInit()가 반환한 INIT_PARAMETERS_INCORRECT 매개변수 집합은 테스트에 적합하지 않은 것으로 간주되며 유전적 최적화 중 다음 모집단을 얻는 데 사용되지 않습니다. "폐기된" 파라미터 세트가 너무 많으면 최적의 EA 파라미터를 검색할 때 결과가 잘못될 수 있습니다. 검색 알고리즘은 최적화 기준 함수가 스무스하며 입력 파라미터의 전체 수에 대한 공백이 없다고 가정합니다.
- OnInit()가 INIT_FAILED를 반환하는 경우, 이는 테스트를 시작할 수 없으며 EA가 에이전트의 메모리에서 언로드됨을 의미합니다. 새 파라미터 세트를 사용하여 다음 패스를 수행하기 위해 EA가 다시 로드됩니다. TesterStop() 호출에 비해 다음 최적화 패스를 시작하는 데 훨씬 더 많은 시간이 소요됩니다.

EA에 대한 샘플 OnInit() 함수

```
//--- 입력 매개 변수
input int      ma_period=20; // 이동 평균 기간

//--- EA에 사용되는 지표의 핸들
int indicator_handle;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ma_period 유효성 검증
   if(ma_period<=0)
     {
      PrintFormat("유효하지 않은 ma_period 입력 값: %d",ma_period);
      return (INIT_PARAMETERS_INCORRECT);
     }
//--- 최적화 중
   if(MQLInfoInteger(MQL_OPTIMIZATION))
     {
      //--- 에이전트에 사용 가능한 RAM 확인
      int available_memory_mb=TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
      if(available_memory_mb<2000)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("테스트 에이전트에 사용할 메모리가 부족함: %d MB",
                     available_memory_mb);
         return (INIT_AGENT_NOT_SUITABLE);
        }
     }
//--- 지표 검증
   indicator_handle=iCustom(_Symbol,_Period,"My_Indicator",ma_period);
   if(indicator_handle==INVALID_HANDLE)
     {
      PrintFormat("My_Indicator 핸들을 생성하지 못했습니다. Error code %d",
                  GetLastError());
      return (INIT_FAILED);
     }
//--- EA 초기화 성공
   return(INIT_SUCCEEDED);
  }
```

더 보기

OnDeinit, Event handling functions, Program running, Client terminal events, Initialization of variables, Creating and deleting objects

© 2000-2025, MetaQuotes Ltd.

OnDeinit

이 함수는 Deinit 이벤트가 발생할 때 지표 및 EA로 호출됩니다. 이것은 실행 중인 MQL5 프로그램의 초기화를 해제하는 데 사용됩니다.

```
void  OnDeinit(
   const int  reason          // 근거 코드 초기화 해제
   );
```

Parameters

reason

[in] 근거 코드 초기화 해제.

반환 값

반환값 없음

참고

다음과 같은 경우 EA 및 지표에 대해 Deinit 이벤트가 생성됩니다:

- 심볼 또는 차트 기간의 변경으로 인해 mql5 프로그램이 다시 초기화되기 전;
- 입력 변경으로 인한 재초기화 전;
- mql5 프로그램 언로드 전.

근거 파라미터는 다음 값을 가질 수 있습니다:

Constant	Value	Description
REASON_PROGRAM	0	EA가 ExpertRemove() 함수 호출 작동을 중지했습니다
REASON_REMOVE	1	차트에서 프로그램이 제거됨
REASON_RECOMPILE	2	프로그램이 재컴파일됨
REASON_CHARTCHANGE	3	심볼이나 차트 기간이 변경됨
REASON_CHARTCLOSE	4	차트가 닫힘
REASON_PARAMETERS	5	입력이 유저에 의해 변경됨
REASON_ACCOUNT	6	계정 설정의 변경으로 인해 다른 계정이 활성화되었거나 거래 서버에 다시 연결됨
REASON_TEMPLATE	7	적용된 다른 차트 템플릿
REASON_INITFAILED	8	OnInit() 핸들러가 0이 아닌 값을 반환
REASON_CLOSE	9	터미널 닫힘

EA 초기화해제 근거 코드는 UninitializeReason() 함수 또는 사전 정의된 _UninitReason 변수에서 수신할 수 있습니다.

EA용 샘플 OnInit() 및 OnDeinit() 함수

```
input int fake_parameter=3;       // 무효 파라미터
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 프로그램이 컴파일된 빌드 번호를 가져옵니다
   Print(__FUNCTION__," Build #",__MQLBUILD__);
//--- 재설정 근거 코드는 OnInit()에서도 얻을 수 있습니다
   Print(__FUNCTION__," EA 재설정 중에 초기화 해제 근거 코드를 수신할 수 있습니다.");
//--- 초기화 해제 근거 코드를 가져오는 첫 번째 방법
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- 초기화 해제 근거 코드를 가져오는 두 번째 방법
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 초기화 해제 근거 코드를 가져오는 첫 번째 방법
   Print(__FUNCTION__," Deinitialization reason code = ",reason);
//--- 초기화 해제 근거 코드를 가져오는 두 번째 방법
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- 초기화 해제 근거 코드를 가져오는 세 번째 방법
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
  }
//+------------------------------------------------------------------+
//| 초기화 해제 근거 코드에 대한 텍스트 설명을 반환 |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="계정이 변경되었습니다";break;
      case REASON_CHARTCHANGE:
         text="심볼 또는 타임프레임이 변경되었습니다";break;
      case REASON_CHARTCLOSE:
         text="차트가 닫혔습니다";break;
      case REASON_PARAMETERS:
         text="입력 매개변수가 변경되었습니다";break;
      case REASON_RECOMPILE:
         text="프로그램 "+__FILE__+" 이 컴파일되었습니다";break;
      case REASON_REMOVE:
         text="프로그램 "+__FILE__+" 이 차트에서 제거되었습니다";break;
```

© 2000-2025, MetaQuotes Ltd.

```
      case REASON_TEMPLATE:
         text="차트에 새 템플릿이 적용되었습니다";break;
      default:text="또 다른 근거";
     }
//---
   return text;
  }
```

더 보기

OnInit, Event handling functions, Program running, Client terminal events, Uninitialization reason codes, Visibility scope and lifetime of variables, Creating and deleting objects

© 2000-2025, MetaQuotes Ltd.

OnTick

이 함수는 새 값을 처리하기 위해 NewTick 이벤트가 발생할 때 EA로 호출됩니다.

```
void  OnTick(void);
```

반환 값

반환값 없음

참고

NewTick 이벤트는 EA가 첨부된 차트의 기호에 대한 새로운 틱을 수신하는 즉시 EA에 대해서만 생성됩니다. 커스텀 지표 또는 스크립트에서 OnTick() 기능을 정의해도 새Tick 이벤트가 생성되지 않으므로 의미가 없습니다.

Tick 이벤트는 EA에 대해서만 생성되지만, NewTick 외에 EA에 대해서도 Timer, BookEvent 및 ChartEvent 이벤트가 생성되므로 EA에서 OnTick() 함수를 사용할 필요는 없습니다.

모든 이벤트는 영수증 순서대로 하나씩 처리됩니다. 대기열에 이미 NewTick 이벤트가 포함되어 있거나 이 이벤트가 처리 단계에 있는 경우 새 NewTick 이벤트가 mql5 애플리케이션 대기열에 추가되지 않습니다.

NewTick 이벤트는 자동 거래 활성화 여부에 관계없이 생성됩니다(AutoTrading 버튼). 비활성화된 자동 거래란 EA로부터의 거래 요청 전송만 금지하는 것을 의미합니다. EA 작업은 중지되지 않습니다.

AutoTrading 버튼을 눌러 자동 거래를 비활성화해도 OnTick() 함수의 현재 실행이 중단되지는 않습니다.

EA의 예제, OnTick() 함수에 전체 거래 논리 포함

```
//+------------------------------------------------------------------+
//|                                                   TradeByATR.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "EA 거래 예시 ₩폭발적₩ 캔들 방향"
#property description "\"폭발적₩ 캔들의 몸체 크기가 k*ATR을 초과합니다"
#property description "\"리버스₩ 파라미터가 시그널 방향을 리버스합니다"

input double lots=0.1;        // 롯 단위의 볼륨
input double kATR=3;          // ATR에서의 시그널 캔들 길이
input int    ATRperiod=20;    // ATR 지표 기간
input int    holdbars=8;      // 포지션을 유지할 막대 수
input int    slippage=10;     // 허용가능 저하
input bool   revers=false;    // 시그널을 반전시킬까요?
input ulong  EXPERT_MAGIC=0;  // EA's MagicNumber
//--- ATR 지표 핸들 보관용
int atr_handle;
//--- 여기에 마지막 ATR 값과 캔들 바디를 저장할 것입니다.
```

© 2000-2025, MetaQuotes Ltd.

```
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 글로벌 변수 최적화
   last_atr=0;
   last_body=0;
//--- 알맞은 볼륨 설정
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- ATR 지표 핸들 생성
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: iATR 생성 실패, 에러 코드 %d",__FUNCTION__,GetLastError());
      return(INIT_FAILED);
     }
//--- EA 초기화 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- EA 작업 종료 코드 통지
   Print(__FILE__,": 초기화해제 근거 코드 = ",reason);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 거래 시그널
   static int signal=0; // +1은 매수 시그널을, -1은 매도 시그널을 의미합니다
//--- 'holdbars' 막대 보다 이전에 오픈된 포지션을 확인 후 마감합니다
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- 새 막대 점검
   if(isNewBar())
     {
      //--- 시그널 유무를 확인
      signal=CheckSignal();
     }
//--- netting 포지션이 오픈한 경우 시그널을 스킵합니다 - 시그널 마감까지 대기하십시오
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       signal=0;
       return; // NewTick 이벤트 핸들러를 종료하고 새 막대가 나타나기 전에 마켓에 진입하지 마십
      }
//--- 헷징 계정의 경우, 각 포지션은 별도로 보유 및 마감합니다
   if(signal!=0)
     {
      //--- 매수 시그널
      if(signal>0)
        {
         PrintFormat("%s: 매수 시그널! Revers=%s",__FUNCTION__,string(revers));
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- 매도 시그널
      if(signal<0)
        {
         PrintFormat("%s: 매도 시그널! Revers=%s",__FUNCTION__,string(revers));
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- OnTick 함수 종료
  }
//+------------------------------------------------------------------+
//| 새 거래 신호 점검                                                |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0은 시그널이 없음을 의미합니다
   int res=0;
//--- 끝에서 두번째의 완성된 막대에서 ATR 값 획득 (막대 인덱스는 2)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- 최근에 마감된 막대의 데이터를 MqlRates 타입 배열로 가져옵니다.
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- 최근 완성된 막대에서 바 몸체 크기를 계산
         last_body=bar[0].close-bar[0].open;
         //--- 최근 바 몸체(인덱스 1)이 이전 ATR 값(인덱스 2)을 초과하면 거래 시그널이 수신됩니
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // 상향 캔들에 대해선 양의 값
        }
      else
         PrintFormat("%s: 최근 막대 수령에 실패! Error",__FUNCTION__,GetLastError());
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      PrintFormat("%s: ATR 지표 값 수신 실패! Error",__FUNCTION__,GetLastError());
//--- 역거래 모드가 활성화된 경우
   res=revers?-res:res;  // 필요한 경우 시그널을 역방향으로(1이 아닌 -1을 반환하고 그 반대로도
//--- 거래 시그널 값을 반환
   return (res);
  }
//+------------------------------------------------------------------+
//|  새 막대가 나타나면 'true'를 반환                                  |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // 현재 막대의 오픈 시간 저장
//--- 0바의 오픈 시간 획득
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- 오픈 시간이 변경된 경우, 새 막대가 도착한 것입니다
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- 로그에서의 새 막대의 오픈 시간에 데이터 표시
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- 새 막대를 열고 메시지를 표시
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- 최근 틱의 데이터 획득
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- 마지막 틱 시간 표시(밀리초)
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- 새로운 막대가 있습니다
      return (true);
     }
//--- 새로운 막대가 없습니다
   return (false);
  }
//+------------------------------------------------------------------+
//| 지정된 볼륨으로 시장 가격으로 구매                                 |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- 시장 가격에 매수
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 지정된 수량으로 시장 가격으로 판매                                |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- 시장 가격에 매도
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| 막대에서 유지 시간별로 포지션 마감                                |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // 오픈 포지션의 수
//--- 오픈 포지션 반복
   for(int i=total-1; i>=0; i--)
     {
      //--- 포지션 매개 변수
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- MagicNumber와 심볼이 일치하는 동안 포지션의 수명이 이미 큰 경우
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // 올바른 메시지 형식을 위해 텍스트 대소문자를 낮춤
         PrintFormat("클로즈 포지션 #%I64u %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- 주문 유형을 설정하고 거래 요청을 전송
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| 거래 요청을 준비 후 전송                                         |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
  {
//--- 구조 선언 및 초기화
   MqlTradeRequest request={0};
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlTradeResult  result={0};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- 요청 파라미터
   request.action   =TRADE_ACTION_DEAL;                        // 거래 작업 유형
   request.position =pos_ticket;                               // 마감의 경우 포지션 티켓
   request.symbol   =Symbol();                                 // 심볼
   request.volume   =volume;                                   // 볼륨
   request.type     =type;                                     // 주문 유형
   request.price    =price;                                    // 거래 가격
   request.deviation=slip;                                     // 가격에서 허용 가능한 편차
   request.magic    =magicnumber;                              // 주문 Magi cNumber
//--- 요청 전송
   if(!OrderSend(request,result))
     {
      //--- 실패에 대한 데이터 표시
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- 성공적 작업 통지
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

더 보기

Event handling functions, Program running, Client terminal events, OnTimer, OnBookEvent, OnChartEvent

© 2000-2025, MetaQuotes Ltd.

OnCalculate

함수는 가격 데이터 변경을 처리하기 위해 Calculate 이벤트가 발생할 때 지표에서 호출됩니다. 두 가지 함수 유형이 있습니다. 이 중 하나만 단일 지표 내에서 사용할 수 있습니다.

데이터 배열을 기반으로 계산

```
int  OnCalculate(
   const int        rates_total,       // price[] 배열 크기
   const int        prev_calculated,   // 이전 호출의 처리 막대 수
   const int        begin,             // price[] 배열의 인덱스 번호의 의미 있는 데이터는
   const double&    price[]            // 계산할 값의 배열
   );
```

현재 시간 프레임 시계열을 기반으로 계산

```
int  OnCalculate(
   const int        rates_total,       // 입력 시계열 크기
   const int        prev_calculated,   // 이전 호출의 처리 막대 수
   const datetime&  time[],            // 시간 배열
   const double&    open[],            // 배열 열기
   const double&    high[],            // 고가 배열
   const double&    low[],             // 저가 배열
   const double&    close[],           // 종가 배열
   const long&      tick_volume[],     // 틱 볼륨 배열
   const long&      volume[],          // 실수 볼륨 배열
   const int&       spread[]           // 스프레드 배열
   );
```

매개변수

rates_total

[in] 계산 지표에서 사용할 수 있는 price[] 배열 또는 입력 시리즈의 크기. 두 번째 기능 유형에서 매개변수 값은 차트에서 시작되는 막대 수에 해당합니다.

prev_calculated

[in] 이전 호출 중 OnCalculate() 기능으로 반환된 값을 포함. 이 기능은 이전 기능 실행 이후 변경되지 않은 막대를 건너 뛰도록 설계되었습니다.

시작

[in] price[] 배열의 의미 있는 인덱스 값부터 시작합니다. 그러면 올바른 값이 없는 누락 또는 초기 데이터를 건너뛸 수 있습니다.

price[]

[in] 계산을 위한 값의 배열. 가격 시계열 또는 계산된 지표 버퍼 중 하나를 price[] 배열로 전달할 수 있습니다. 계산을 위한 전달된 데이터 유형은 _AppliedTo 사전 정의된 변수를 사용하여 정의할 수 있습니다.

time[]

[in] 막대 열기 시간 값이 있는 배열.

open[]

© 2000-2025, MetaQuotes Ltd.

[in] 시가 값이 포함된 배열.

high[]

[in] 고가 값이 포함된 배열.

low[]

[in] 저가 값이 포함된 배열.

close[]

[in] 종가 값이 포함된 배열.

tick_volume[]

[in] 틱 볼륨 값이 포함된 배열.

volume[]

[in] 거래 볼륨 값이 포함된 배열.

spread[]

[in] 막대를 위한 스프레드 값이 포함된 배열.

반환 값

int 유형 값은 다음 함수 호출 시 *prev_calculated* 매개변수로 전달됩니다.

참고

OnCalculate() 기능이 0이면 클라이언트 터미널의 DataWindow에 아무런 지표 값도 표시되지 않습니다.

OnCalculate() 함수의 마지막 호출 이후 가격 데이터가 변경된 경우 (더 깊은 내역이 로드되었거나 내역 공백이 채워진 경우) *prev_calculated* 입력 매개변수의 값은 터미널 자체에서 0으로 설정됩니다.

time[], *open[]*, *high[]*, *low[]*, *close[]*, *tick_volume[]*, *volume[]* 및 *spread[]* 배열에서 인덱싱 방향을 정의하려면 ArrayGetAsSeries() 기능을 호출. 기본값을 사용하지 않으려면 배열과 함께 사용할 ArraySetAsSeries() 기능을 호출.

첫 번째 기능 유형을 사용하는 경우 사용자가 지표를 실행할 때 매개변수 탭에서 price[] 배열로 필요한 시계열 또는 지표를 선택합니다. 이렇게 하려면 "적용 대상" 필드의 드롭다운 목록에서 필요한 요소를 지정.

다른 mql5 프로그램에서 사용자 지정 지표 값을 가져오려면 iCustom() 기능이 사용됩니다. 후속 작업에 대한 지표 핸들을 반환합니다. 필요한*price []* 배열 또는 다른 지표의 핸들을 지정할 수도 있습니다. 이 매개변수는 사용자 지정 지표의 입력 변수 목록에서 마지막 매개변수를 전달해야 합니다.

OnCalculate() 기능에 의해 반환되는 값과 *prev_calculated* 초 입력 매개변수 사이의 연결을 사용해야 합니다. 기능을 호출할 때 *prev_calculated* 매개변수에는 이전 호출 중 OnCalculate() 함수가 반환한 값이 포함됩니다. 따라서 이 기능의 이전 실행 이후 변경되지 않은 막대에 대한 반복적인 계산을 방지하기 위해 사용자 지정 지표를 계산하기 위한 리소스 절약 알고리즘을 구현할 수 있습니다.

샘플 지표

```
//+------------------------------------------------------------------+
//|                                          OnCalculate_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "샘플 모멘텀 지표 계산"

//---- 지표 설정
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_type1   DRAW_LINE
#property indicator_color1  Blue
//---- 입력
input int MomentumPeriod=14; // 계산 주기
//---- 지표 버퍼
double    MomentumBuffer[];
//--- 계산 주기를 저장을 위한 글로벌 변수
int       IntPeriod;
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 입력 매개변수를 확인
   if(MomentumPeriod<0)
     {
      IntPeriod=14;
      Print("주기 매개변수 값이 잘못되었습니다. 다음 값은 계산에서 사용됩니다 ",IntPeriod);
     }
   else
      IntPeriod=MomentumPeriod;
//---- 버퍼
   SetIndexBuffer(0,MomentumBuffer,INDICATOR_DATA);
//---- DataWindow 및 하위 창에 표시할 지표 이름
   IndicatorSetString(INDICATOR_SHORTNAME,"Momentum"+"("+string(IntPeriod)+")");
//--- 그리기가 시작되는 막대의 인덱스를 설정
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,IntPeriod-1);
//--- 0.0을 그려지지 않은 빈 값으로 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//--- 지표 정확도 표시
   IndicatorSetInteger(INDICATOR_DIGITS,2);
  }
//+------------------------------------------------------------------+
//|  모멘텀 지표 계산                                                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,     // price[] 배열 크기
                const int prev_calculated, // 이전에 처리한 막대의 수
```

© 2000-2025, MetaQuotes Ltd.

```
                const int begin,           // 중요한 데이터가 시작되는 지점
                const double &price[])     // 처리를 위한 값 배열
  {
//--- 계산을 위한 초기 포지션
   int StartCalcPosition=(IntPeriod-1)+begin;
//---- 계산 데이터가 충분하지 않은 경우
   if(rates_total<StartCalcPosition)
      return(0);  // 0 값을 사용하여 종료 - 지표가 계산되지 않음
//--- 정확한 그리기 시작
   if(begin>0)
      PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,StartCalcPosition+(IntPeriod-1));
//--- 시작 포지션을 정의하는 계산 시작
   int pos=prev_calculated-1;
   if(pos<StartCalcPosition)
      pos=begin+IntPeriod;
//--- 주 계산 루프
   for(int i=pos;i<rates_total && !IsStopped();i++)
      MomentumBuffer[i]=price[i]*100/price[i-IntPeriod];
//--- OnCalculate 실행이 완료되었습니다. 후속 호출에 대해 새 prev_calculated 값을 반환
   return(rates_total);
  }
```

추가 참조

ArrayGetAsSeries, ArraySetAsSeries, iCustom, 이벤트 처리 기능, 프로그램 작동, 클라이언트 터미널 이벤트, 시계열 및 지표에 액세스

© 2000-2025, MetaQuotes Ltd.

OnTimer

이 함수는 터미널에서 일정한 시간 간격으로 생성되는 Timer 이벤트 동안 EA 단위로 호출됩니다.

```
void  OnTimer(void);
```

반환 값

반환값 없음

참고

Timer 이벤트는 클라이언트 터미널에서 주기적으로 EA에 대해 생성되며, 클라이언트 터미널은 EventSetTimer() 함수를 사용하여 타이머를 활성화합니다. 일반적으로 이 함수는 OnInit() 함수에서 호출됩니다. EA가 작동을 멈추면 EventKillTimer()를 사용하여타이머를 제거해야 하며, 이는 일반적으로 OnDeinit() 함수에서 호출됩니다.

각 Expert Advisor와 지표는 이 타이머에서만 이벤트를 수신하는 자체 타이머로 작동합니다. mql5 애플리케이션 종료 중에 타이머가 생성되었지만 EventKillTimer() 함수에 의해 비활성화되지 않은 경우 타이머가 강제로 폐기됩니다..

초당 한 번 이상 타이머 이벤트를 수신해야 하는 경우 EventSetMillisecondTimer()를 사용하여 고해상도 타이머를 생성하십시오.

일반적으로 타이머 시간이 줄어들면 타이머 이벤트 핸들러가 더 자주 호출되므로 테스트 시간이 늘어납니다. 실시간 모드에서 작업할 경우 하드웨어 제한으로 인해 타이머 이벤트가 10-16밀리초 내에 1회 이상 생성되지 않습니다.

각 프로그램에 대해 타이머를 하나만 시작할 수 있습니다. 각 mql5 애플리케이션 및 차트에는 새로 도착한 모든 이벤트가 배치되는 자체 이벤트 대기열이 있습니다. 대기열에 이미 Timer 이벤트가 포함되어 있거나 이 이벤트가 처리 단계에 있는 경우 새 Timer 이벤트가 mql5 애플리케이션 대기열에 추가되지 않습니다.

샘플 EA OnTimer() 핸들러 포함

```
//+------------------------------------------------------------------+
//|                                                OnTimer_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "거래 서버 시간 계산에 타이머 사용 예제"
#property description "EA는 주말 전에 거래 주 말에 실행하는 것이 좋습니다."
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 1초의 타이머를 생성
   EventSetTimer(1);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 작업 종료 후 타이머 제거
   EventKillTimer();

  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| 타이머 함수                                                      |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- OnTimer()의 첫 호출 시간
   static datetime start_time=TimeCurrent();
//--- 첫 번째 OnTimer() 호출 중 트레이드 서버 시간
   static datetime start_tradeserver_time=0;
//--- 거래 서버 시간 계산
   static datetime calculated_server_time=0;
//--- 현지 PC 시간
   datetime local_time=TimeLocal();
//--- 현재 예상 거래 서버 시간
   datetime trade_server_time=TimeTradeServer();
//--- 어떤 이유로 서버 시간을 알 수 없는 경우 미리 종료
   if(trade_server_time==0)
      return;
//--- 초기 트레이드 서버 값이 아직 설정되지 않은 경우
   if(start_tradeserver_time==0)
     {
      start_tradeserver_time=trade_server_time;
      //--- 거래 서버의 계산값 설정
      Print(trade_server_time);
      calculated_server_time=trade_server_time;
     }
   else
     {
      //--- OnTimer()의 첫 호출 시간을 증가시킴
```

© 2000-2025, MetaQuotes Ltd.

```
      if(start_tradeserver_time!=0)
         calculated_server_time=calculated_server_time+1;;
      }
//---
   string com=StringFormat("                  Start time: %s\r\n",TimeToString(start_t
   com=com+StringFormat("                  Local time: %s\r\n",TimeToString(local_time
   com=com+StringFormat("TimeTradeServer time: %s\r\n",TimeToString(trade_server_time,
   com=com+StringFormat(" EstimatedServer time: %s\r\n",TimeToString(calculated_server
//--- 차트에 모든 카운터 값 표시
   Comment(com);
  }
```

더 보기

EventSetTimer, EventSetMillisecondTimer, EventKillTimer, GetTickCount, GetMicrosecondCount, Client terminal events

© 2000-2025, MetaQuotes Ltd.

OnTrade

이 함수는 Trade 이벤트가 발생할 때 EA로 호출됩니다. 이 기능은 순서, 위치 및 거래 목록의 변경을 처리하기 위한 것입니다.

```
void  OnTrade(void);
```

반환 값

반환값 없음

참고

OnTrade()는 Expert Advisors 전용으로 호출됩니다. 이름과 유형이 동일한 함수를 추가하는 경우에도 지표 및 스크립트에서 사용되지 않습니다.

모든 거래 행위(보류 주문, 오프닝/클로징, 정지, 주문 활성화 등)의 경우, 주문 및 거래 내역 및/또는 포지션 목록과 현재 주문 목록을 적절하게 변경합니다.

주문을 처리할 때 거래 서버는 터미널에 들어오는 Trade 이벤트에 대한 메시지를 보냅니다. 주문 및 거래 내역 관련 데이터를 검색하려면 먼저 HistorySelect()를 사용하여 거래 내역 요청을 수행해야 합니다.

다음과 같은 경우 서버에서 거래 이벤트를 생성합니다:

- 활성화 주문 변경,
- 포지션 변경,
- 딜 변경,
- 거래 이력 변경.

각 Trade 이벤트는 하나 또는 여러 거래 요청의 결과로 나타날 수 있습니다. 거래 요청은 OrderSend() 또는 OrderSendAsync()을 사용하여 서버로 전송됩니다. 각 요청은 여러 가지 거래 이벤트를 유발할 수 있습니다. "하나의 요청당 하나의 거래 이벤트"라는 문장은 꼭 맞다고 하기 어렵습니다. 이벤트의 처리는 여러 단계로 수행될 수 있으며 각 작업의 주문 상태, 포지션 및 거래 기록이 변경될 수 있기 때문입니다.

OnTrade() 핸들러는 알맞은 OnTradeTransaction() 호출 뒤에 호출됩니다. 일반적으로 OnTrade () 및 OnTradeTransaction () 요청 수에는 정확한 상관관계가 없습니다. 하나의 OnTrade() 요청은 하나 이상의 OnTradeTransaction 호출에 해당합니다.

샘플 EA OnTrade() 핸들러 포함

```
//+------------------------------------------------------------------+
//|                                                OnTrade_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

input    int days=7;             // 며칠간 거래의 뎁스 내역
//--- 거래 이력의 한계를 글로벌 범위로 설정
```

© 2000-2025, MetaQuotes Ltd.

```
datetime     start;              // 캐시 내 거래 이력 시작 날짜
datetime     end;                // 캐시 내 거래 이력 종료 날짜
//--- 글로벌 카운터
int          orders;             // 활성 주문 수
int          positions;          // 오픈 포지션 수
int          deals;              // 거래 내역 캐시의 거래 수
int          history_orders;     // 거래 내역 캐시의 주문 수
bool         started=false;      // 역관계의 플래그

//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   end=TimeCurrent();
   start=end-days*PeriodSeconds(PERIOD_D1);
   PrintFormat("로드할 기록의 제한: start - %s, end - %s",
               TimeToString(start),TimeToString(end));
   InitCounters();
//---
   return(0);
  }
//+------------------------------------------------------------------+
//|  포지션, 주문 및 거래 카운터의 초기화                            |
//+------------------------------------------------------------------+
void InitCounters()
  {
   ResetLastError();
//--- 로드 이력
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Failed to load history from %s to %s to cache. Error code: %d",
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- 현재 값 획득
   orders=OrdersTotal();
   positions=PositionsTotal();
   deals=HistoryDealsTotal();
   history_orders=HistoryOrdersTotal();
   started=true;
   Print("주문, 포지션 및 거래 카운터가 초기화되었습니다");
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| 거래 이벤트가 도착하면 호출됩니다                                |
//+------------------------------------------------------------------+
void OnTrade()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| 거래 및 내역 변경의 처리 예                                      |
//+------------------------------------------------------------------+
void SimpleTradeProcessor()
  {
   end=TimeCurrent();
   ResetLastError();
//--- 지정된 간격에서 프로그램 캐시로 거래 기록 다운로드
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Failed to load history from %s to %s to cache. Error code: %d",
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- 현재 값 획득
   int curr_orders=OrdersTotal();
   int curr_positions=PositionsTotal();
   int curr_deals=HistoryDealsTotal();
   int curr_history_orders=HistoryOrdersTotal();
//--- 활성 주문 건수가 변경되었는지 점검
   if(curr_orders!=orders)
     {
      //--- 활성 주문 수가 변경되었습니다
      PrintFormat("주문 건수가 변경되었습니다. 이전 값은 %d, 현재 값은 %d",
                  orders,curr_orders);
      //--- 값 업데이트
      orders=curr_orders;
     }
//--- 오픈 포지션 수의 변화
   if(curr_positions!=positions)
     {
      //--- 오픈 포지션 수가 변경되었습니다
      PrintFormat("포지션 수가 변경되었습니다. 이전 값은 %d, 현재 값은 %d",
                  positions,curr_positions);
      //--- 값 업데이트
      positions=curr_positions;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 거래 내역 캐시의 딜 수 변경
   if(curr_deals!=deals)
     {
      //--- 거래 기록 캐시의 딜 수가 변경되었습니다
      PrintFormat("딜 수가 변경되었습니다. 이전 값은 %d, 현재 값은 %d",
                  deals,curr_deals);
      //--- 값 업데이트
      deals=curr_deals;
     }
//--- 거래 기록 캐시의 기록 주문 수 변경
   if(curr_history_orders!=history_orders)
     {
      //--- 거래 기록 캐시의 기록 주문 수가 변경되었습니다
      PrintFormat("기록 내 주문 수가 변경되었습니다. 이전 값은 %d, 현재 값은 %d",
                  history_orders,curr_history_orders);
     //--- 값 업데이트
     history_orders=curr_history_orders;
     }
//--- 캐시에서 요청할 거래 기록의 한도를 변경할 필요가 있는지 확인
   CheckStartDateInTradeHistory();
  }
//+------------------------------------------------------------------+
//|  거래 내역 요청 시작 날짜 변경         |
//+------------------------------------------------------------------+
void CheckStartDateInTradeHistory()
  {
//--- 초기 간격, 지금 당장 작업을 시작할 경우
   datetime curr_start=TimeCurrent()-days*PeriodSeconds(PERIOD_D1);
//--- 거래 내역의 시작 제한이 없어지지 않도록 하십시오
//--- 예정일보다 1일 이상 경과
   if(curr_start-start>PeriodSeconds(PERIOD_D1))
     {
      //--- 캐시에 로드할 기록 시작 날짜를 수정
      start=curr_start;
      PrintFormat("로드할 거래 기록의 새 시작 제한: 시작 => %s",
                  TimeToString(start));
      //--- 이제 업데이트된 간격에 대한 거래 기록을 다시 로드합니다
      HistorySelect(start,end);
      //--- 그 이상의 비교를 위해 딜과 주문 카운터를 수정
      history_orders=HistoryOrdersTotal();
      deals=HistoryDealsTotal();
     }
  }
//+------------------------------------------------------------------+
/* 샘플 결과:
  로드할 기록의 한계: 시작 - 2018.07.16 18:11, 끝 - 2018.07.23 18:11
  주문, 포지션 및 거래 카운터가 성공적으로 초기화되었습니다.
  주문 수가 변경되었습니다. 이전 값 0, 현재 값 1
```

© 2000-2025, MetaQuotes Ltd.

```
   주문 수가 변경되었습니다. 이전 값 0, 현재 값 1
   포지션 수가 변경되었습니다. 이전 값 0, 현재 값 1
   딜 수가 변경되었습니다. 이전 값 0, 현재 값 1
   내역의 주문 수가 변경되었습니다. 이전 값 0, 현재 값 1
*/
```

더 보기

OrderSend, OrderSendAsync, OnTradeTransaction, Client terminal events

© 2000-2025, MetaQuotes Ltd.

OnTradeTransaction

이 함수는 TradeTransaction 이벤트가 발생할 때 EA 단위로 호출됩니다. 이 함수는 거래 요청 실행 결과를 처리하기 위한 것입니다.

```
void  OnTradeTransaction()
   const MqlTradeTransaction&    trans,     // 거래 트랜잭션 구조
   const MqlTradeRequest&        request,   // 요청 구조
   const MqlTradeResult&         result     // 반응 구조
   );
```

Parameters

trans

[in] 거래 계정에서 수행된 트랜잭션을 설명하는 MqlTradeTransaction 유형 변수.

request

[in] 거래 요청을 설명하는 MqlTradeRequest 유형 변수. 여기에는 TRADE_TRANSACTION_REQUEST 유형 트랜잭션에 대한 값만 포함됩니다.

result

[in] 트랜잭션으로 이어진 거래 요청의 실행 결과를 포함하는 MqlTradeResult 유형 변수. 여기에는 TRADE_TRANSACTION_REQUEST 유형 트랜잭션에 대한 값만 포함됩니다.

반환 값

반환값 없음

참고

OnTradeTransaction()은 다음과 같은 경우 트레이드 서버가 터미널에 보내는 TradeTransaction 이벤트를 처리하기 위해 호출됩니다:

- OrderSend()/OrderSendAsync() 함수 및 후속 실행을 사용하여 MQL5 프로그램의 거래 요청 전송;
- GUI 및 그 후속 실행을 통해 수동으로 거래 요청 전송;
- 서버에서 보류 중인 명령 및 중지 명령의 활성화;
- 트레이드 서버 쪽에서 작업을 수행.

트랜잭션 유형에 대한 데이터는 *유형* 필드, *trans* 변수에 포함됩니다. 트레이드 트랜잭션의 유형은 ENUM_TRADE_TRANSACTION_TYPE 열거값에 설명되어 있습니다:

- TRADE_TRANSACTION_ORDER_ADD – 새 활성 주문 추가
- TRADE_TRANSACTION_ORDER_UPDATE – 기존 주문 변경
- TRADE_TRANSACTION_ORDER_DELETE – 활성 주문 목록에서 주문 삭제
- TRADE_TRANSACTION_DEAL_ADD – 이력에 딜 추가
- TRADE_TRANSACTION_DEAL_UPDATE – 이력의 딜 변경
- TRADE_TRANSACTION_DEAL_DELETE – 이력에서 딜 삭제
- TRADE_TRANSACTION_HISTORY_ADD – 실행 또는 취소의 결과로 내역에 주문 추가
- TRADE_TRANSACTION_HISTORY_UPDATE – 주문 이력에서 주문 변경
- TRADE_TRANSACTION_HISTORY_DELETE – 주문 내역에서 주문 삭제
- TRADE_TRANSACTION_POSITION – 거래 실행과 무관한 포지션 변경
- TRADE_TRANSACTION_REQUEST – 거래 요청이 서버에서 처리되었으며 처리 결과가 수신되었음을 알리는 알림.

© 2000-2025, MetaQuotes Ltd.

TRADE_TRANSACTION_REQUEST 타입의 거래를 처리할 때는, OnTradeTransaction() 함수의 두 번째, 세 번째 변수 – *요청* 및 *결과* – 를 분석하여 추가 정보를 얻어야 합니다.

매수 거래 요청을 전송하면 거래 계정의 연쇄 트레이드 트랜잭션으로 이어집니다. 1) 요청을 처리하도록 수락, 2) 계정에 대해 적절한 구매 주문 작성, 3) 주문이 실행됨, 4) 실행된 주문이 활성 주문 리스트에서 제거, 5) 후속 트랜잭션 기록 추가, 6) 이력에 추가, 7) 새 포지션이 생성됩니다. 이 모든 단계는 트레이드 트랜잭션입니다. 이러한 각 트랜잭션이 터미널에 도착하는 것은 TradeTransaction 이벤트입니다. 이러한 거래의 터미널 도착 우선순위는 보장되지 않습니다. 따라서 거래 알고리즘을 개발할 때 한 그룹의 트랜잭션이 차례로 도착할 것이라고 예상해서는 안 됩니다.

EA의 OnTradeTransaction() 핸들러가 트랜잭션을 처리하면 터미널은 들어오는 거래 트랜잭션을 처리합니다. 따라서 OnTradeTransaction() 작업 과정에서 거래 계정 상태가 변경될 수 있습니다. 예를 들어, MQL5 프로그램이 새 주문 추가를 처리하는 동안 실행하거나 열려 있는 주문 목록에서 삭제하고 기록으로 이동할 수 있습니다. 프로그램에 이러한 모든 이벤트가 통지됩니다.

1024개의 요소들로 이루어진 거래 대기 길이. OnTradeTransaction()이 다른 트랜잭션을 너무 오랫동안 처리하면 이전 트랜잭션이 대기열의 새 트랜잭션으로 대체될 수 있습니다.

OnTrade() 핸들러는 적절한 OnTradeTransaction() 호출 뒤에 호출됩니다. 일반적으로 OnTrade () 및 OnTradeTransaction () 요청 수에는 정확한 상관관계가 없습니다. 한 번의 OnTrade() 호출은 하나 이상의 OnTradeTransaction 호출에 해당합니다.

각 Trade 이벤트는 하나 또는 여러 거래 요청의 결과로 나타날 수 있습니다. 거래 요청은OrderSend() 또는 OrderSendAsync()을 사용하여 서버로 전송됩니다. 각 요청은 여러 가지 거래 이벤트를 유발할 수 있습니다. "하나의 요청당 하나의 거래 이벤트"라는 문장은 꼭 맞다고 하기 어렵습니다. 이벤트의 처리는 여러 단계로 수행될 수 있으며 각 작업의 주문 상태, 포지션 및 거래 기록이 변경될 수 있기 때문입니다.

샘플 EA OnTradeTransaction() 핸들러 포함

```
//+------------------------------------------------------------------+
//|                                    OnTradeTransaction_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "TradeTransaction 이벤트의 샘플 리스너"
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   PrintFormat("LAST PING=%.f ms",
               TerminalInfoInteger(TERMINAL_PING_LAST)/1000.);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| TradeTransaction 함수                                            |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//---
   static int counter=0;   // OnTradeTransaction() 호출의 카운터
   static uint lasttime=0; // OnTradeTransaction() 최근 호출 시간
//---
   uint time=GetTickCount();
//--- 마지막 거래가 1초 이상 전에 수행된 경우,
   if(time-lasttime>1000)
     {
      counter=0; // 그러면 이것은 새로운 거래 작업이고 카운터는 재설정될 수 있습니다
      if(IS_DEBUG_MODE)
         Print(" 새 거래 작업");
     }
   lasttime=time;
   counter++;
   Print(counter,". ",__FUNCTION__);
//--- 거래 요청 실행 결과
   ulong            lastOrderID   =trans.order;
   ENUM_ORDER_TYPE  lastOrderType =trans.order_type;
   ENUM_ORDER_STATE lastOrderState=trans.order_state;
//--- 트랜잭션이 수행된 심볼의 이름
   string trans_symbol=trans.symbol;
//--- 트랜잭션 유형
   ENUM_TRADE_TRANSACTION_TYPE  trans_type=trans.type;
   switch(trans.type)
     {
      case  TRADE_TRANSACTION_POSITION:   // 포지션 변경
        {
         ulong pos_ID=trans.position;
         PrintFormat("MqlTradeTransaction: Position  #%I64u %s modified: SL=%.5f TP=%.
                     pos_ID,trans_symbol,trans.price_sl,trans.price_tp);
        }
      break;
      case TRADE_TRANSACTION_REQUEST:     // 거래 요청 전송
         PrintFormat("MqlTradeTransaction: TRADE_TRANSACTION_REQUEST");
         break;
      case TRADE_TRANSACTION_DEAL_ADD:    // 거래 추가
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          ulong          lastDealID   =trans.deal;
          ENUM_DEAL_TYPE lastDealType =trans.deal_type;
          double         lastDealVolume=trans.volume;
          //--- 외부 시스템의 거래 ID - 거래소에서 할당한 티켓
          string Exchange_ticket="";
          if(HistoryDealSelect(lastDealID))
             Exchange_ticket=HistoryDealGetString(lastDealID,DEAL_EXTERNAL_ID);
          if(Exchange_ticket!="")
             Exchange_ticket=StringFormat("(Exchange deal=%s)",Exchange_ticket);

          PrintFormat("MqlTradeTransaction: %s deal #%I64u %s %s %.2f lot   %s",EnumToS
                      lastDealID,EnumToString(lastDealType),trans_symbol,lastDealVolume
         }
       break;
       case TRADE_TRANSACTION_HISTORY_ADD: // 내역에 주문 추가
         {
          //--- 외부 시스템의 주문 ID - 거래소에서 할당한 티켓
          string Exchange_ticket="";
          if(lastOrderState==ORDER_STATE_FILLED)
            {
             if(HistoryOrderSelect(lastOrderID))
                Exchange_ticket=HistoryOrderGetString(lastOrderID,ORDER_EXTERNAL_ID);
             if(Exchange_ticket!="")
                Exchange_ticket=StringFormat("(Exchange ticket=%s)",Exchange_ticket);
            }
          PrintFormat("MqlTradeTransaction: %s order #%I64u %s %s %s   %s",EnumToString
                      lastOrderID,EnumToString(lastOrderType),trans_symbol,EnumToString
         }
       break;
       default: // 기타 트랜잭션
         {
          //--- 외부 시스템의 주문 ID - 거래소에서 할당한 티켓
          string Exchange_ticket="";
          if(lastOrderState==ORDER_STATE_PLACED)
            {
             if(OrderSelect(lastOrderID))
                Exchange_ticket=OrderGetString(ORDER_EXTERNAL_ID);
             if(Exchange_ticket!="")
                Exchange_ticket=StringFormat("Exchange ticket=%s",Exchange_ticket);
            }
          PrintFormat("MqlTradeTransaction: %s order #%I64u %s %s   %s",EnumToString(ti
                      lastOrderID,EnumToString(lastOrderType),EnumToString(lastOrderSta
         }
       break;
      }
//--- 주문 티켓
   ulong orderID_result=result.order;
   string retcode_result=GetRetcodeID(result.retcode);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(orderID_result!=0)
      PrintFormat("MqlTradeResult: order #%d retcode=%s ",orderID_result,retcode_resul
//---
  }
//+------------------------------------------------------------------+
//| 숫자 응답 코드를 문자열 니모닉으로 변환                          |
//+------------------------------------------------------------------+
string GetRetcodeID(int retcode)
  {
   switch(retcode)
     {
      case 10004: return("TRADE_RETCODE_REQUOTE");             break;
      case 10006: return("TRADE_RETCODE_REJECT");              break;
      case 10007: return("TRADE_RETCODE_CANCEL");              break;
      case 10008: return("TRADE_RETCODE_PLACED");              break;
      case 10009: return("TRADE_RETCODE_DONE");                break;
      case 10010: return("TRADE_RETCODE_DONE_PARTIAL");        break;
      case 10011: return("TRADE_RETCODE_ERROR");               break;
      case 10012: return("TRADE_RETCODE_TIMEOUT");             break;
      case 10013: return("TRADE_RETCODE_INVALID");             break;
      case 10014: return("TRADE_RETCODE_INVALID_VOLUME");      break;
      case 10015: return("TRADE_RETCODE_INVALID_PRICE");       break;
      case 10016: return("TRADE_RETCODE_INVALID_STOPS");       break;
      case 10017: return("TRADE_RETCODE_TRADE_DISABLED");      break;
      case 10018: return("TRADE_RETCODE_MARKET_CLOSED");       break;
      case 10019: return("TRADE_RETCODE_NO_MONEY");            break;
      case 10020: return("TRADE_RETCODE_PRICE_CHANGED");       break;
      case 10021: return("TRADE_RETCODE_PRICE_OFF");           break;
      case 10022: return("TRADE_RETCODE_INVALID_EXPIRATION");  break;
      case 10023: return("TRADE_RETCODE_ORDER_CHANGED");       break;
      case 10024: return("TRADE_RETCODE_TOO_MANY_REQUESTS");   break;
      case 10025: return("TRADE_RETCODE_NO_CHANGES");          break;
      case 10026: return("TRADE_RETCODE_SERVER_DISABLES_AT");  break;
      case 10027: return("TRADE_RETCODE_CLIENT_DISABLES_AT");  break;
      case 10028: return("TRADE_RETCODE_LOCKED");              break;
      case 10029: return("TRADE_RETCODE_FROZEN");              break;
      case 10030: return("TRADE_RETCODE_INVALID_FILL");        break;
      case 10031: return("TRADE_RETCODE_CONNECTION");          break;
      case 10032: return("TRADE_RETCODE_ONLY_REAL");           break;
      case 10033: return("TRADE_RETCODE_LIMIT_ORDERS");        break;
      case 10034: return("TRADE_RETCODE_LIMIT_VOLUME");        break;
      case 10035: return("TRADE_RETCODE_INVALID_ORDER");       break;
      case 10036: return("TRADE_RETCODE_POSITION_CLOSED");     break;
      default:
         return("TRADE_RETCODE_UNKNOWN="+IntegerToString(retcode));
         break;
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

더 보기

OrderSend, OrderSendAsync, OnTradeTransaction, Trade request structure, Trade transaction structure, Trade transaction types, Trade operation types, Client terminal events

© 2000–2025, MetaQuotes Ltd.

OnBookEvent

이 기능은 BookEvent 이벤트가 발생할 때 지표와 EA로 호출됩니다. Depth of Market 변화에 대처하기 위한 것입니다.

```
void  OnBookEvent(
   const string&  symbol         // 심볼
   );
```

매개변수

심볼

[in] BookEvent가 도착한 심볼의 이름

반환 값

값 반환이 없음

참고

모든 기호에 대한 BookEvent 이벤트를 받으려면 MarketBookAdd() 기능을 사용하여 구독하면 이 기호에 대한 이벤트를 받을 수 있습니다. 특정 기호에 대한 BookEvent 수신 신청을 취소하려면 MarketBookRelease() 기능을 호출.

BookEvent는 전체 차트 내에서 브로드캐스트됩니다. 즉, 차트의 애플리케이션이 MarketBookAdd 기능을 사용하여 BookEvent를 구독하는 경우 다른 모든 지표와 EA가 동일한 차트에서 시작되고 OnBookEvent() 처리기도 이 이벤트를 수신하게 됩니다. 따라서 OnBookEvent() 처리기에 전달된 심볼 이름을 *심볼* 매개변수로 분석해야 합니다.

동일한 차트에서 실행 중인 모든 애플리케이션에 대해 심볼별로 정렬된 별도의 BookEvent 카운터가 제공됩니다. 즉, 각 차트에 서로 다른 심볼에 대한 구독이 여러 개 있을 수 있으며 각 심볼에 대해 카운터가 제공됩니다. BookEvent를 구독 및 구독 취소하면 한 차트 내에서 지정된 심볼에 대한 구독 카운터가 변경됩니다. 즉, 동일한 심볼이지만 구독 카운터 값이 서로 다른 두 개의 BookEvent 인접 차트가 있을 수 있습니다.

초기 구독 카운터 값은 0입니다. 각 MarketBookAdd() 호출 시 차트에서 지정된 기호에 대한 구독 카운터 1씩 증가합니다(MarketBookAdd()의 차트 심볼 및 심볼은 일치하지 않아도 됨). MarketBookRelease()를 호출하면 차트에서 지정된 심볼에 대한 구독 카운터가 1만큼 줄어듭니다. 모든 기호에 대한 BookEvent 이벤트는 카운터가 0이 될 때까지 차트 내에서 브로드캐스트됩니다. 따라서 MarketBookAdd () 호출이 포함된 각 MQL5 프로그램은 각 작업이 끝날 때 MarketBookRelease ()를 사용하여 각 심볼에 대한 이벤트를 올바르게 구독 취소하는 것이 중요합니다. 이를 위해서는 전체 MQL5 프로그램 수명 동안 각 호출마다 MarketBookAdd() 및 MarketBookRelease() 호출 횟수가 균등해야 합니다. 프로그램 내에서 플래그 또는 사용자 지정 구독 카운터를 사용하면 BookEvent 이벤트를 안전하게 작업할 수 있으며 동일한 차트 내의 타사 프로그램에서 이 이벤트를 가져오기 위한 구독을 사용하지 않도록 설정할 수 있습니다.

BookEvent 이벤트는 건너뛰지 않으며 이전 BookEvent 처리를 아직 완료하지 않은 경우에도 항상 대기열에 배치됩니다.

예

```
//+------------------------------------------------------------------+
//|                                        OnBookEvent_Sample.mq5 |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com/en/articles/2635"
#property version   "1.00"
#property description "OnBookEvent()를 사용하여 시장 수준 업데이트/교체율 측정 예"
#property description "코드는 다음 문서에서 가져왔습니다. https://www.mql5.com/en/articles
//--- 매개변수 입력
input ulong ExtCollectTime   =30;  // 테스트 시간(초)
input ulong ExtSkipFirstTicks=10;  // 시작할 때 건너뛴 틱 수
//--- BookEvent 이벤트 구독 플래그
bool book_subscribed=false;
//--- 시장 수준으로부터의 요청을 수락하기 위한 배열
MqlBookInfo  book[];
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 기능                                             |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 시작 표시
   Comment(StringFormat("첫 %I64u 틱이 도착할 때까지 기다리는 중",ExtSkipFirstTicks));
   PrintFormat("첫 %I64u 틱이 도착할 때까지 기다리는 중",ExtSkipFirstTicks);
//--- 시장 수준 브로드캐스트 가능
   if(MarketBookAdd(_Symbol))
     {
      book_subscribed=true;
      PrintFormat("%s: MarketBookAdd(%s) 함수가 true 반환함",__FUNCTION__,_Symbol);
     }
   else
      PrintFormat("%s: MarketBookAdd(%s) 함수가 false 반환함! GetLastError()=%d",__FUNC
//--- 성공적인 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 초기화 해제 엑스퍼트                                             |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 초기화 취소 이유 코드 표시
   Print(__FUNCTION__,": 초기화 해제 이유 코드 = ",reason);
//--- 시장 수준 이벤트 가져오기를 위해 구독을 취소
   if(book_subscribed)
     {
      if(!MarketBookRelease(_Symbol))
         PrintFormat("%s: MarketBookRelease(%s)가 false 반환! GetLastError()=%d",_Symb
      else
         book_subscribed=false;
```

© 2000-2025, MetaQuotes Ltd.

```
     }
//---
  }
//+------------------------------------------------------------------+
//| BookEvent 기능                                                   |
//+------------------------------------------------------------------+
void OnBookEvent(const string &symbol)
  {
   static ulong starttime=0;              // 테스트 시작 시간
   static ulong tickcounter=0;            // 시장 수준 업데이트 카운터
//--- 직접 구독해야 시장 수준 이벤트를 진행할 수 있습니다
   if(!book_subscribed)
      return;
//--- 특정 기호에 대한 업데이트만의 수
   if(symbol!=_Symbol)
      return;
//--- 첫 번째 체크 표시를 건너 뛰고 대기열을 지우고 준비
   tickcounter++;
   if(tickcounter<ExtSkipFirstTicks)
      return;
//--- 시작 시간을 기억
   if(tickcounter==ExtSkipFirstTicks)
      starttime=GetMicrosecondCount();
//--- 시장 수준 데이터를 요청
   MarketBookGet(symbol,book);
//--- 언제 멈추나요?
   ulong endtime=GetMicrosecondCount()-starttime;
   ulong ticks  =1+tickcounter-ExtSkipFirstTicks;
// 테스트 시작 후 시간(마이크로초)이 얼마나 지났습니까?
   if(endtime>ExtCollectTime*1000*1000)
     {
      PrintFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec ",ticks,endtime/1000.0
      ExpertRemove();
      return;
     }
//--- 코멘트 필드에 카운터를 표시
   if(endtime>0)
      Comment(StringFormat("%I64u ticks for %.1f seconds: %.1f ticks/sec ",ticks,endti
  }
```

추가 참조

MarketBookAdd, MarketBookRelease, MarketBookGet, OnTrade, OnTradeTransaction, OnTick, 이벤트 핸들링 기능, 프로그램 작동, 클라이언트 터미널 이벤트

© 2000-2025, MetaQuotes Ltd.

OnChartEvent

이 함수는 ChartEvent 이벤트가 발생할 때 지표와 EA로 호출됩니다. 이 함수는 사용자 또는 MQL5 프로그램에 의해 수행된 차트 변경 사항을 처리하기 위한 것입니다.

```
void  OnChartEvent()
   const int       id,       // 이벤트 ID
   const long&     lparam,   // long 타입 이벤트 파라미터
   const double&   dparam,   // double 타입 이벤트 파라미터
   const string&   sparam    // string 타입 이벤트 파라미터
   );
```

Parameters

id

[in] Event ID from the ENUM_CHART_EVENT 열거값.

lparam

[in] long 타입 이벤트 파라미터

dparam

[in] double 타입 이벤트 파라미터

sparam

[in] string 타입 이벤트 파라미터

반환 값

반환값 없음

참고

사전 정의된 OnChartEvent() 함수를 사용하여 처리할 수 있는 이벤트 유형에는 11가지가 있습니다. 커스텀 이벤트를 위해 CHARTEVENT_CUSTOM 에서부터 CHARTEVENT_CUSTOM_LAST 까지 65535개 ID가 제공됩니다. 커스텀 이벤트를 생성하려면 EventChartCustom() 함수를 사용하십시오.

ENUM_CHART_EVENT 열거값의 간단한 이벤트 설명:

- CHARTEVENT_KEYDOWN — 차트 창이 포커스에 있을 때 키보드의 키를 누름;
- CHARTEVENT_MOUSE_MOVE — 마우스 및 마우스 버튼 클릭 이동 (CHART_EVENT_MOUSE_MOVE가 차트에서 참인 경우);
- CHARTEVENT_OBJECT_CREATE — 그래픽 개체 생성 (CHART_EVENT_OBJECT_CREATE가 차트에서 참인 경우);
- CHARTEVENT_OBJECT_CHANGE — 속성 대화 상자를 통해 개체 속성 변경;
- CHARTEVENT_OBJECT_DELETE — 그래픽 개체 삭제 (CHART_EVENT_OBJECT_DELETE가 차트에서 참인 경우);
- CHARTEVENT_CLICK — 차트에서 클릭;
- CHARTEVENT_OBJECT_CLICK — 차트에 속하는 그래픽 개체를 마우스 클릭;
- CHARTEVENT_OBJECT_DRAG — 마우스를 사용하여 그래픽 개체 끌기;
- CHARTEVENT_OBJECT_ENDEDIT — 그래픽 개체의 편집 입력 상자에서 텍스트 편집 완료 (OBJ_EDIT);
- CHARTEVENT_CHART_CHANGE — 차트 변경;

© 2000-2025, MetaQuotes Ltd.

- CHARTEVENT_CUSTOM+n — 사용자 지정 이벤트 ID (여기서 n은 0 ~ 65535 범위). CHARTEVENT_CUSTOM_LAST 에는 마지막으로 허용된 커스텀 이벤트 ID(CHARTEVENT_CUSTOM+65535)가 포함되어 있습니다.

모든 MQL5 programs은 어플리케이션의 메인 스레드 이외의 스레드에서 작동합니다. 메인 어플리케이션 스레드는 모든 Windows 시스템 메시지를 처리하고, 그 결과 자체 어플리케이션에 대한 Windows 메시지를 생성합니다. 예를 들어, 차트(WM_MOUSE_MOVE event)에서 마우스를 이동하면 응용 프로그램 창의 후속 렌더링을 위한 여러 시스템 메시지가 생성되며 차트에서 시작된 experts 및 지표에 내부 메시지가 전송됩니다. EA 또는 지표가 마우스 이동 이벤트를 이미 수신한 상태에서 주 애플리케이션 스레드가 아직 WM_PAINT 시스템 메시지를 처리하지 않은(따라서 수정된 차트를 렌더링하지 않은) 상황이 발생할 수 있습니다. 이 경우 차트 속성 CHART_FIRST_VISIBLE_BAR은 차트를 렌더링한 후에만 변경됩니다.

각 이벤트 유형에 대해 OnChartEvent() 함수의 입력에는 해당 이벤트를 처리하는 데 필요한 특정 값이 포함됩니다. 표에는 파라미터를 통해 전달된 이벤트와 값이 나와 있습니다.

Event	'id' 파라미터 값	'lparam' 파라미터 값	'dparam' 파라미터 값	'sparam' 파라미터 값
Keystroke event	CHARTEVENT_KEYDOWN	pressed button code	키를 누른 상태로 유지하는 동안 생성된 키 누름 횟수	키보드 키의 상태를 설명하는 비트 마스크의 문자열 값
마우스 이벤트 (CHART_EVENT_MOUSE_MOVE가 차트에서 참인 경우)	CHARTEVENT_MOUSE_MOVE	X 좌표	Y 좌표	마우스 키의 상태를 설명하는 비트 마스크의 문자열 값
마우스 휠 이벤트 (CHART_EVENT_MOUSE_WHEEL 가 차트에서 참인 경우)	CHARTEVENT_MOUSE_WHEEL	커서의 키와 마우스 버튼, X 및 Y 좌표에 대한 플래그. 다음 예제의 설명을 보십시오	마우스 휠 스크롤의 델타 값	—
그래픽 개체 생성 (CHART_EVENT_OBJECT_CREATE 가 차트에서 참인 경우)	CHARTEVENT_OBJECT_CREATE	—	—	생성된 그래픽 개체의 이름
속성 대화 상자를 통해 개체 속성 변경	CHARTEVENT_OBJECT_CHANGE	—	—	변경된 그래픽 개체의 이름
그래픽 개체 제거 (CHART_EVENT_OBJECT_DELETE 가 차트에서 참인 경우)	CHARTEVENT_OBJECT_DELETE	—	—	제거된 그래픽 개체의 이름
차트에서 마우스 클릭	CHARTEVENT_CLICK	X 좌표	Y 좌표	—

© 2000-2025, MetaQuotes Ltd.

Event	'id' 파라미터 값	'lparam' 파라미터 값	'dparam' 파라미터 값	'sparam' 파라미터 값
그래픽 개체에 마우스 클릭	CHARTEVENT_OBJECT_CLICK	X 좌표	Y 좌표	이벤트가 발생한 그래픽 개체의 이름
마우스를 사용하여 그래픽 개체 이동	CHARTEVENT_OBJECT_DRAG	—	—	이동된 그래픽 개체의 이름
"입력 필드" 그래픽 객체 입력 상자에서 텍스트 편집 완료	CHARTEVENT_OBJECT_ENDEDIT	—	—	텍스트 편집이 완료된 "입력 필드" 그래픽 개체의 이름
속성 대화창을 통해 차트 크기 조정 또는 차트 속성 수정	CHARTEVENT_CHART_CHANGE	—	—	—
커스텀 이벤트(N번호)	CHARTEVENT_CUSTOM+N	EventChartCustom() 함수에 의해 정의된 값	EventChartCustom() 함수에 의해 정의된 값	EventChartCustom() 함수에 의해 정의된 값

샘플 차트 이벤트 리스너:

```
//+------------------------------------------------------------------+
//|                                            OnChartEvent_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "샘플 차트 이벤트 리스너 및 커스텀 이벤트 생성기"
//--- 서비스 키 ID
#define KEY_NUMPAD_5       12
#define KEY_LEFT           37
#define KEY_UP             38
#define KEY_RIGHT          39
#define KEY_DOWN           40
#define KEY_NUMLOCK_DOWN   98
#define KEY_NUMLOCK_LEFT  100
#define KEY_NUMLOCK_5     101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP    104
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- CHARTEVENT_CUSTOM 상수 값 표시
   Print("CHARTEVENT_CUSTOM=",CHARTEVENT_CUSTOM);
//---
   Print("Launched the EA ",MQLInfoString(MQL5_PROGRAM_NAME));
//--- 수신한 차트 개체의 생성 이벤트의 플래그를 설정
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- 수신한 차트 개체의 제거 이벤트의 플래그를 설정
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- 마우스 휠 스크롤 메시지 실행
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- 차트 속성 강제 업데이트로 이벤트 처리 준비 상태 보장
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 커스텀 이벤트 생성을 위한 틱 카운터
   static int tick_counter=0;
//--- 누적 틱을 이 값으로 나눔
   int simple_number=113;
//---
   tick_counter++;
//--- 체크 카운터가 simple_number의 배수인 경우 커스텀 이벤트 전송
   if(tick_counter%simple_number==0)
     {
      //--- 커스텀 이벤트 ID를 0에서부터 65535까지 내에서 생성
      ushort custom_event_id=ushort(tick_counter%65535);
      //---  매개 변수를 채우는 커스텀 이벤트 전송
      EventChartCustom(ChartID(),custom_event_id,tick_counter,SymbolInfoDouble(Symbol
      //--- 예제 결과 분석을 위해 로그에 추가
      Print(__FUNCTION__,": Sent a custom event ID=",custom_event_id);
     }
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent 함수                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- 키 입력
   if(id==CHARTEVENT_KEYDOWN)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      switch((int)lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("Pressed KEY_NUMLOCK_LEFT");   break;
         case KEY_LEFT:          Print("Pressed KEY_LEFT");           break;
         case KEY_NUMLOCK_UP:    Print("Pressed KEY_NUMLOCK_UP");     break;
         case KEY_UP:            Print("Pressed KEY_UP");             break;
         case KEY_NUMLOCK_RIGHT: Print("Pressed KEY_NUMLOCK_RIGHT"); break;
         case KEY_RIGHT:         Print("Pressed KEY_RIGHT");          break;
         case KEY_NUMLOCK_DOWN:  Print("Pressed KEY_NUMLOCK_DOWN");   break;
         case KEY_DOWN:          Print("Pressed KEY_DOWN");           break;
         case KEY_NUMPAD_5:      Print("Pressed KEY_NUMPAD_5");       break;
         case KEY_NUMLOCK_5:     Print("Pressed KEY_NUMLOCK_5");      break;
         default:                Print("목록에 없는 키를 누름");
        }
     }
//--- 차트에서 좌클릭
   if(id==CHARTEVENT_CLICK)
      Print("차트에서 마우스 클릭 좌표: x = ",lparam,"  y = ",dparam);
//--- 그래픽 개체 클릭
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("이름이 지정된 개체의 마우스 버튼 클릭 '"+sparam+"'");
//--- 개체 제거
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("다음으로 이름이 지명된 개체 제거 ",sparam);
//--- 생성된 개체
   if(id==CHARTEVENT_OBJECT_CREATE)
      Print("다음으로 이름이 지명된 개체 생성 ",sparam);
//--- 변경된 개체
   if(id==CHARTEVENT_OBJECT_CHANGE)
      Print("다음으로 이름이 지명된 개체 변경 ",sparam);
//--- 객체가 이동했거나 앵커 포인트 좌표가 변경됨
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("다음으로 이름이 지정된 개체의 앵커 포인트 변경 ",sparam);
//--- Edit(편집) 그래픽 개체의 입력 필드에서 텍스트 변경
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("편집 개체에서 텍스트가 변경됨 ",sparam,"  id=",id);
//--- 마우스 이동 이벤트
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- 이 이벤트에 대한 마우스 버튼 및 휠 상태 고려
      int flg_keys = (int)(lparam>>32);          // Ctrl 및 Shift 키와 마우스 버튼의 상태
      int x_cursor = (int)(short)lparam;         // 마우스 휠 이벤트가 발생한 위치의 X 좌표
      int y_cursor = (int)(short)(lparam>>16);   // 마우스 휠 이벤트가 발생한 위치의 Y 좌표
      int delta    = (int)dparam;                // 마우스 스크롤의 총 값이 +120 또는 -12
      //--- 플래그 처리
      string str_keys="";
      if((flg_keys&0x0001)!=0)
```

© 2000-2025, MetaQuotes Ltd.

```
         str_keys+="LMOUSE ";
      if((flg_keys&0x0002)!=0)
         str_keys+="RMOUSE ";
      if((flg_keys&0x0004)!=0)
         str_keys+="SHIFT ";
      if((flg_keys&0x0008)!=0)
         str_keys+="CTRL ";
      if((flg_keys&0x0010)!=0)
         str_keys+="MMOUSE ";
      if((flg_keys&0x0020)!=0)
         str_keys+="X1MOUSE ";
      if((flg_keys&0x0040)!=0)
         str_keys+="X2MOUSE ";

      if(str_keys!="")
         str_keys=", keys='"+StringSubstr(str_keys,0,StringLen(str_keys)-1)+"'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
//--- 속성 대화 상자 창을 사용하여 차트 크기 조정 또는 차트 속성 수정 이벤트
   if(id==CHARTEVENT_CHART_CHANGE)
      Print("차트 크기 또는 속성 변경");
//--- 커스텀 이벤트
   if(id>CHARTEVENT_CUSTOM)
      PrintFormat("Custom event ID=%d, lparam=%d, dparam=%G, sparam=%s",id,lparam,dpar
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // mouse left
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // mouse right
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // mouse middle
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // mouse first X key
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // mouse second X key
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift key
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control key
   return(res);
  }
```

더 보기

EventChartCustom, Types of chart events, Event handling functions, Program running, Client terminal events

© 2000-2025, MetaQuotes Ltd.

OnTester

테스트 후 필요한 작업을 수행하기 위해 Tester 이벤트가 발생하면 Expert Advisor에서 이 기능을 호출합니다.

```
double  OnTester(void);
```

반환 값

테스트 결과 평가를 위한 커스텀 기준 최적화 값.

참고

OnTester() 함수는 EA를 테스트할 때만 사용할 수 있으며, 주로 입력 파라미터를 최적화할 때 'Custom max' 기준으로 사용되는 값을 계산하기 위한 것입니다.

유전 최적화 중에는 한 세대 내의 결과 정렬이 내림차순으로 수행됩니다. 즉, 최적화 기준 관점에서 값이 가장 높은 결과가 가장 우수하다고 판단됩니다. 이러한 정렬에 대한 최악의 값은 마지막에 배치되고 나중에 삭제됩니다. 그러므로, 그들은 다음 세대를 형성하는데 참여하지 않습니다.

따라서 OnTester() 함수를 통해 테스트 결과 보고서를 직접 생성하고 저장할 수 있을 뿐만 아니라 최적화 프로세스를 제어하여 거래 전략의 최적의 파라미터를 찾을 수 있습니다.

다음은 커스텀 기준 최적화를 계산하는 **예제**입니다. 아이디어는 밸런스 그래프의 선형 회귀를 계산하는 것입니다. 밸런스 그래프를 이용한 전략 최적화 및 "밸런스 + 최대 Sharpe Ratio" 기준과 결과 비교 기고글에 설명되어 있습니다.

```
//+------------------------------------------------------------------+
//|                                                OnTester_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "OnTester() 핸들러 포함 샘플 EA"
#property description "커스텀 최적화 기준으로, "
#property description "밸런스 그래프 선형 회귀의 비율"
#property description "평균 제곱근 편차로 나눈 값이 반환됩니다."
//--- 거래 작업을 위한 등급을 포함
#include <Trade\Trade.mqh>
//--- EA 입력 파라미터
input double Lots              = 0.1;      // 볼륨
input int    Slippage          = 10;       // 허용가능 저하
input int    MovingPeriod      = 80;       // 이동 평균 기간
input int    MovingShift       = 6;        // 이동 평균 시프트
//--- 글로벌 변수
int    IndicatorHandle=0;  // 지표 처리
bool   IsHedging=false;    // 계정의 플래그
CTrade trade;              // 거래 작업 수행용
//---
#define EA_MAGIC 18052018
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 포지션 오프닝 조건 점검                                          |
//+------------------------------------------------------------------+
void CheckForOpen(void)
  {
   MqlRates rt[2];
//--- 새 막대의 시작점에서만 거래
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
//--- 틱 볼륨
   if(rt[1].tick_volume>1)
      return;
//--- 이동 평균 값 수신
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- 신호 유무 점검
   ENUM_ORDER_TYPE signal=WRONG_VALUE;
//--- 캔들이 더 고점에서 열렸지만, 이동평균보다 다 저점에서 닫혔음
   if(rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=ORDER_TYPE_BUY;    // 매수 시그널
   else // 캔들이 더 저점에서 열렸지만, 이동평균보다 더 고점에서 닫혔음
     {
      if(rt[0].open<ma[0] && rt[0].close>ma[0])
         signal=ORDER_TYPE_SELL;// 매도 시그널
     }
//--- 추가 점검
   if(signal!=WRONG_VALUE)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
        {
         double price=SymbolInfoDouble(_Symbol,signal==ORDER_TYPE_SELL ? SYMBOL_BID:SY
         trade.PositionOpen(_Symbol,signal,Lots,price,0,0);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| 포지션 클로징 조건 점검                                          |
//+------------------------------------------------------------------+
void CheckForClose(void)
  {
   MqlRates rt[2];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 새 막대의 시작점에서만 거래
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," failed, no history");
      return;
     }
   if(rt[1].tick_volume>1)
      return;
//--- 이동 평균 값 수신
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("CopyBuffer from iMA failed, no data");
      return;
     }
//--- PositionSelect()를 사용하여 이전에 이미 포지션이 선택되었음
   bool signal=false;
   long type=PositionGetInteger(POSITION_TYPE);
//--- 캔들이 더 고점에서 열렸지만, 이동평균보다 더 저점에서 닫혔음 - 숏 포지션 마감
   if(type==(long)POSITION_TYPE_SELL && rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=true;
//--- 캔들이 더 저점에서 열렸지만, 이동평균보다 더 고점에서 닫혔음 - 롱 포지션 마감
   if(type==(long)POSITION_TYPE_BUY && rt[0].open<ma[0] && rt[0].close>ma[0])
      signal=true;
//--- 추가 점검
   if(signal)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
         trade.PositionClose(_Symbol,Slippage);
     }
//---
  }
//+------------------------------------------------------------------+
//| 계정 유형을 고려하여 포지션 선택: Netting 또는 Hedging |
//+------------------------------------------------------------------+
bool SelectPosition()
  {
   bool res=false;
//--- Hedging 계정용 포지션 선택
   if(IsHedging)
     {
      uint total=PositionsTotal();
      for(uint i=0; i<total; i++)
        {
         string position_symbol=PositionGetSymbol(i);
         if(_Symbol==position_symbol && EA_MAGIC==PositionGetInteger(POSITION_MAGIC))
           {
            res=true;
            break;
```

© 2000-2025, MetaQuotes Ltd.

```
            }
         }
      }
//--- Netting 계정용 포지션 선택
   else
     {
      if(!PositionSelect(_Symbol))
         return(false);
      else
         return(PositionGetInteger(POSITION_MAGIC)==EA_MAGIC); //---check Magic numbe
     }
//--- 실행 결과
   return(res);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- 거래 유형 설정: Netting 또는 Hedging
   IsHedging=((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOU
//--- 올바른 포지션 제어를 위해 개체 초기화
   trade.SetExpertMagicNumber(EA_MAGIC);
   trade.SetMarginMode();
   trade.SetTypeFillingBySymbol(Symbol());
   trade.SetDeviationInPoints(Slippage);
//--- 이동 평균 지표 생성
   IndicatorHandle=iMA(_Symbol,_Period,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE);
   if(IndicatorHandle==INVALID_HANDLE)
     {
      printf("iMA 지표 생성 에러");
      return(INIT_FAILED);
     }
//--- ok
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick(void)
  {
//--- 이미 포지션이 오픈된 경우 마감 조건을 점검하십시오
   if(SelectPosition())
      CheckForClose();
// 포지션 오프닝 조건 점검
   CheckForOpen();
//---
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 테스터 함수                                                      |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- 사용자 정의 기준 최적화 값(높을수록 좋음)
   double ret=0.0;
//--- 거래 결과를 배열로 얻기
   double array[];
   double trades_volume;
   GetTradeResultsToArray(array,trades_volume);
   int trades=ArraySize(array);
//--- 거래가 10번 미만인 경우, 양성 결과가 나오지 않습니다
   if(trades<10)
      return (0);
//--- 거래당 평균 실적
   double average_pl=0;
   for(int i=0;i<ArraySize(array);i++)
      average_pl+=array[i];
   average_pl/=trades;
//--- 단일 테스트 모드에 대한 메시지를 표시합니다
   if(MQLInfoInteger(MQL_TESTER) && !MQLInfoInteger(MQL_OPTIMIZATION))
      PrintFormat("%s: Trades=%d, Average profit=%.2f",__FUNCTION__,trades,average_pl);
//--- 이익 그래프에 대한 선형 회귀 비율을 계산
   double a,b,std_error;
   double chart[];
   if(!CalculateLinearRegression(array,chart,a,b))
      return (0);
//--- 회귀선에서 차트 편차의 오차를 계산
   if(!CalculateStdError(chart,a,b,std_error))
      return (0);
//--- 표준 편차에 대한 추세 이익 비율을 계산
   ret=(std_error == 0.0) ? a*trades : a*trades/std_error;
//--- 사용자 정의 기준 최적화 값 반환
   return(ret);
  }
//+------------------------------------------------------------------+
//| 거래에서의 수익/손실 배열 얻기                                   |
//+------------------------------------------------------------------+
bool GetTradeResultsToArray(double &pl_results[],double &volume)
  {
//--- 전체 거래 내역 요청
   if(!HistorySelect(0,TimeCurrent()))
      return (false);
   uint total_deals=HistoryDealsTotal();
   volume=0;
//--- 내역에서 거래 수에 따라 마진으로 배열의 초기 크기를 설정
   ArrayResize(pl_results,total_deals);
//--- 거래 결과를 확정하는 거래 카운터 - 수익 또는 손실
   int counter=0;
```

© 2000-2025, MetaQuotes Ltd.

```
   ulong ticket_history_deal=0;
//--- 모든 거래 살펴보기
   for(uint i=0;i<total_deals;i++)
     {
      //--- 딜 선택
      if((ticket_history_deal=HistoryDealGetTicket(i))>0)
        {
         ENUM_DEAL_ENTRY deal_entry  =(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket_hi
         long            deal_type   =HistoryDealGetInteger(ticket_history_deal,DEAL_T
         double          deal_profit =HistoryDealGetDouble(ticket_history_deal,DEAL_PR
         double          deal_volume =HistoryDealGetDouble(ticket_history_deal,DEAL_VO
         //--- 저희는 오직 거래 작업에만 관심이 있습니다
         if((deal_type!=DEAL_TYPE_BUY) && (deal_type!=DEAL_TYPE_SELL))
            continue;
         //--- 수익/손실을 확정하는 거래만
         if(deal_entry!=DEAL_ENTRY_IN)
           {
            //--- 거래 결과를 배열에 기록하고 거래량을 늘림
            pl_results[counter]=deal_profit;
            volume+=deal_volume;
            counter++;
           }
        }
     }
//--- 배열의 최종 규모 설정
   ArrayResize(pl_results,counter);
   return (true);
  }
//+------------------------------------------------------------------+
//| 선형 회귀 계산 y=a*x+b                                           |
//+------------------------------------------------------------------+
bool CalculateLinearRegression(double  &change[],double &chartline[],
                               double  &a_coef,double  &b_coef)
  {
//--- 자료가 충분한지 확인
   if(ArraySize(change)<3)
      return (false);
//--- 누적 차트 배열 생성
   int N=ArraySize(change);
   ArrayResize(chartline,N);
   chartline[0]=change[0];
   for(int i=1;i<N;i++)
      chartline[i]=chartline[i-1]+change[i];
//--- 이제 회귀 비율을 계산
   double x=0,y=0,x2=0,xy=0;
   for(int i=0;i<N;i++)
     {
      x=x+i;
      y=y+chartline[i];
```

© 2000-2025, MetaQuotes Ltd.

```
      xy=xy+i*chartline[i];
      x2=x2+i*i;
     }
   a_coef=(N*xy-x*y)/(N*x2-x*x);
   b_coef=(y-a_coef*x)/N;
//---
   return (true);
  }
//+------------------------------------------------------------------+
//|  지정된 a 및 b에 대한 평균 제곱근 편차 계산 오류     |
//+------------------------------------------------------------------+
bool  CalculateStdError(double  &data[],double  a_coef,double  b_coef,double &std_err)
  {
//--- 오차 제곱합
   double error=0;
   int N=ArraySize(data);
   if(N<=2)
      return (false);
   for(int i=0;i<N;i++)
      error+=MathPow(a_coef*i+b_coef-data[i],2);
   std_err=MathSqrt(error/(N-2));
//---
   return (true);
  }
```

더 보기

Testing trading strategies, TesterHideIndicators, Working with optimization results, TesterStatistics, OnTesterInit, OnTesterDeinit, OnTesterPass, MQL_TESTER, MQL_OPTIMIZATION, FileOpen, FileWrite, FileLoad, FileSave

© 2000-2025, MetaQuotes Ltd.

OnTesterInit

전략 테스터에서 최적화하기 전에 필요한 작업을 수행하기 위해 TesterInit 이벤트가 발생하면 EA에서 이 기능을 호출합니다. 두 가지 함수 유형이 있습니다.

결과를 반환하는 버전

```
int  OnTesterInit(void);
```

반환 값

int 타입 값, 0은 최적화가 시작되기 전에 차트에서 EA가 성공적으로 초기화되었음을 의미합니다.

실행 결과를 반환하는 OnTesterInit() 호출은 프로그램 초기화를 허용할 뿐만 아니라 초기 최적화 중지 시 오류 코드를 반환하므로 사용을 권장합니다. INIT_SUCCEEDED (0) 이외의 값이 반환되면 오류가 발생하며 최적화가 시작되지 않습니다.

결과 반환 없는 버전은 이전 코드와의 호환성을 위해서만 유지됩니다. 이용을 권장하지 않습니다

```
void  OnTesterInit(void);
```

참고

TesterInit 이벤트는 전략 테스터에서 EA 최적화가 시작되기 전에 생성됩니다. 이 이벤트에서는 OnTesterDeInit() 또는 OnTesterPass() 이벤트 핸들러가 있는 EA가 별도의 터미널 차트에 자동으로 다운로드됩니다. 여기에는 테스터에 지정된 심볼과 기간이 있습니다.

그러한 이벤트는 TesterInit, TesterDeinit 및 TesterPass 이벤트를 수신하며, Init, Deinit 및 NewTick은 수신하지 않습니다. 따라서 최적화 중 각 패스의 결과를 처리하는 데 필요한 모든 논리는 OnTesterInit (), OnTesterDeinit () 및 OnTesterPass () 핸들러에 구현되어야 합니다.

전략 최적화 중 단일 패스의 결과는 OnTester () 핸들러의 프레임을 통해 FrameAdd () 함수를 사용하여 전달될 수 있습니다.

OnTesterInit() 기능은 최적화를 시작하기 전에 Expert Advisor를 시작하여 추가적인 최적화 결과 처리를 수행하는 데 사용됩니다. 이것은 항상 OnTesterDeinit() 핸들러와 함께 사용됩니다.

OnTesterInit() 실행 시간은 제한되어 있습니다. 이 값을 초과하면 EA가 강제로 중지되고 최적화 자체는 취소됩니다. 테스터 저널에 다음과 같은 메시지가 표시됩니다:

```
Tester OnTesterInit가 너무 오래 작업하고 있습니다. 테스터가 초기화될 수 없습니다.
```

이 예제는 OnTick에서 가져온 것입니다. 최적화 파라미터를 설정하기 위해 OnTesterInit() 핸들러가 추가되었습니다:

```
//+------------------------------------------------------------------+
//|                                          OnTesterInit_Sample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "OnTesterInit() 핸들러 포함 샘플 EA."
#property description "다음의 값과 제한이 있는 "
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "최적화 중 입력 설정"

input double lots=0. 1;        // 롯 단위 볼륨
input double kATR=3;           // ATR에서의 시그널 캔들 길이
input int    ATRperiod=20;     // ATR 지표 기간
input int    holdbars=8;       // 포지션을 유지할 막대 수
input int    slippage=10;      // 허용가능 저하
input bool   revers=false;     // 시그널을 반전시킬까요?
input ulong  EXPERT_MAGIC=0;   // EA's MagicNumber
//--- ATR 지표 핸들 보관용
int atr_handle;
//--- 여기에 마지막 ATR 값과 캔들 바디를 저장할 것입니다.
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//--- 최적화 시작 시간 기억
datetime optimization_start;
//--- 최적화 종료 후 차트에 지속 시간 표시
string report;
//+------------------------------------------------------------------+
//| TesterInit 함수                                                  |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- 최적화할 입력 값을 설정
   ParameterSetRange("lots",false,0.1,0,0,0);
   ParameterSetRange("kATR",true,3.0,1.0,0.3,7.0);
   ParameterSetRange("ATRperiod",true,10,15,1,30);
   ParameterSetRange("holdbars",true,5,3,1,15);
   ParameterSetRange("slippage",false,10,0,0,0);
   ParameterSetRange("revers",true,false,false,1,true);
   ParameterSetRange("EXPERT_MAGIC",false,123456,0,0,0);
   Print("초기값 및 최적화 파라미터 제한이 설정됨");
//--- 최적화 시작 기억
   optimization_start=TimeLocal();
   report=StringFormat("%s: optimization launched at %s",
                       __FUNCTION__,TimeToString(TimeLocal(),TIME_MINUTES|TIME_SECONDS
//--- 차트 및 터미널 저널에 메시지 표시
   Print(report);
   Comment(report);
//---
  }
//+------------------------------------------------------------------+
//| TesterDeinit 함수                                                |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- 최적화 기간
   string log_message=StringFormat("%s: optimization took %d seconds",
```

© 2000-2025, MetaQuotes Ltd.

```
                                    __FUNCTION__,TimeLocal()-optimization_start);
   PrintFormat(log_message);
   report=report+"\r\n"+log_message;
   Comment(report);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 글로벌 변수 최적화
   last_atr=0;
   last_body=0;
//--- 알맞은 볼륨 설정
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- ATR 지표 핸들 생성
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: iATR 생성 실패, 에러 코드 %d",__FUNCTION__,GetLastError());
      return(INIT_FAILED);
     }
//--- EA 초기화 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 거래 시그널
   static int signal=0; // +1은 매수 시그널을, -1은 매도 시그널을 의미합니다
//--- 'holdbars' 막대 보다 이전에 오픈된 포지션을 확인 후 마감합니다
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- 새 막대 점검
   if(isNewBar())
     {
      //--- 시그널 유무를 확인
      signal=CheckSignal();
     }
//--- netting 포지션이 오픈한 경우 시그널을 스킵합니다 - 시그널 마감까지 대기하십시오
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // NewTick 이벤트 핸들러를 종료하고 새 막대가 나타나기 전에 마켓에 진입하지 마십
     }
//--- 헷징 계정의 경우, 각 포지션은 별도로 보유 및 마감합니다
   if(signal!=0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 매수 시그널
       if(signal>0)
         {
          PrintFormat("%s: 매수 시그널! Revers=%s",__FUNCTION__,string(revers));
          if(Buy(trade_lot,slippage,EXPERT_MAGIC))
             signal=0;
         }
       //--- 매도 시그널
       if(signal<0)
         {
          PrintFormat("%s: 매도 시그널! Revers=%s",__FUNCTION__,string(revers));
          if(Sell(trade_lot,slippage,EXPERT_MAGIC))
             signal=0;
         }
      }
//--- OnTick 함수 종료
  }
//+------------------------------------------------------------------+
//| 새 거래 신호 점검                                                |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0은 시그널이 없음을 의미합니다
   int res=0;
//--- 끝에서 두번째의 완성된 막대에서 ATR 값 획득 (막대 인덱스는 2)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- 최근에 마감된 막대의 데이터를 MqlRates 타입 배열로 가져옵니다.
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- 최근 완성된 막대에서 바 몸체 크기를 계산
         last_body=bar[0].close-bar[0].open;
         //--- 최근 바 몸체(인덱스 1)이 이전 ATR 값(인덱스 2)을 초과하면 거래 시그널이 수신됩니
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // 상향 캔들에 대해선 양의 값
        }
      else
         PrintFormat("%s: 최근 막대 수령에 실패! Error",__FUNCTION__,GetLastError());
     }
   else
      PrintFormat("%s: ATR 지표 값 수신 실패! Error",__FUNCTION__,GetLastError());
//--- 역거래 모드가 활성화된 경우
   res=revers?-res:res;  // 필요한 경우 시그널을 역방향으로(1이 아닌 -1을 반환하고 그 반대로도
//--- 거래 시그널 값을 반환
   return (res);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  새 막대가 나타나면 'true'를 반환                                  |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // 현재 막대의 오픈 시간 저장
//--- 0바의 오픈 시간 획득
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- 오픈 시간이 변경된 경우, 새 막대가 도착한 것입니다
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- 로그에서의 새 막대의 오픈 시간에 데이터 표시
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- 새 막대를 열고 메시지를 표시
         PrintFormat("%s: new bar on %s %s opened at %s",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- 최근 틱의 데이터 획득
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() failed, error = ",GetLastError());
         //--- 마지막 틱 시간 표시(밀리초)
         PrintFormat("Last tick was at %s.%03d",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- 새로운 막대가 있습니다
      return (true);
     }
//--- 새로운 막대가 없습니다
   return (false);
  }
//+------------------------------------------------------------------+
//| 지정된 볼륨으로 시장 가격으로 구매                                 |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- 시장 가격에 매수
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| 지정된 수량으로 시장 가격으로 판매                                 |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- 시장 가격에 매도
```

© 2000-2025, MetaQuotes Ltd.

```
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| 막대에서 유지 시간별로 포지션 마감                                |
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // 오픈 포지션의 수
//--- 오픈 포지션 반복
   for(int i=total-1; i>=0; i--)
     {
      //--- 포지션 매개 변수
      ulong  position_ticket=PositionGetTicket(i);
      string position_symbol=PositionGetString(POSITION_SYMBOL);
      ulong  magic=PositionGetInteger(POSITION_MAGIC);
      datetime position_open=(datetime)PositionGetInteger(POSITION_TIME);
      int bars=iBarShift(_Symbol,PERIOD_CURRENT,position_open)+1;

      //--- MagicNumber와 심볼이 일치하는 동안 포지션의 수명이 이미 큰 경우
      if(bars>holdtimebars && magic==magicnumber && position_symbol==_Symbol)
        {
         int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS);
         double volume=PositionGetDouble(POSITION_VOLUME);
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         string str_type=StringSubstr(EnumToString(type),14);
         StringToLower(str_type); // 올바른 메시지 형식을 위해 텍스트 대소문자를 낮춤
         PrintFormat("클로즈 포지션 #%I64u %s %s %.2f",
                     position_ticket,position_symbol,str_type,volume);
         //--- 주문 유형을 설정하고 거래 요청을 전송
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| 거래 요청을 준비 후 전송                                         |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
  {
//--- 구조 선언 및 초기화
   MqlTradeRequest request={0};
   MqlTradeResult  result={0};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- 요청 파라미터
   request.action   =TRADE_ACTION_DEAL;                     // 거래 작업 유형
```

© 2000-2025, MetaQuotes Ltd.

```
   request.position =pos_ticket;                                  // 마감의 경우 포지션 티켓
   request.symbol   =Symbol();                                    // 심볼
   request.volume   =volume;                                      // 볼륨
   request.type     =type;                                        // 주문 유형
   request.price    =price;                                       // 거래 가격
   request.deviation=slip;                                        // 가격에서 허용 가능한 편차
   request.magic    =magicnumber;                                 // 주문 Magi cNumber
//--- 요청 전송
   if(!OrderSend(request,result))
     {
      //--- 실패에 대한 데이터 표시
      PrintFormat("OrderSend %s %s %.2f at %.5f error %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- 성공적 작업 통지
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

더 보기

Testing trading strategies, Working with optimization results, OnTesterDeinit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2025, MetaQuotes Ltd.

OnTesterDeinit

이 함수는 EA 최적화 후 TesterDeinit 이벤트가 발생할 때 EA로 호출됩니다.

```
void  OnTesterDeinit(void);
```

반환 값

반환값 없음

참고

TesterDeinit이벤트는 전략 테스터에서 EA 최적화 종료 후 발생합니다.

최적화 시작 중에 OnTesterDeInit() 또는 OnTesterPass() 이벤트 핸들러가 있는 EA가 별도의 터미널 차트에 자동으로 다운로드됩니다. 여기에는 테스터에 지정된 심볼과 기간이 있습니다. 이 함수는 모든 최적화 결과의 최종 처리를 위해 설계되었습니다.

테스트 에이전트에서 FrameAdd() 함수를 사용하여 전송하는 최적화 프레임은 번들로 제공되며 제공에 시간이 걸릴 수 있습니다. 따라서 TesterPass 이벤트뿐 아니라 모든 프레임이 OnTesterPass()로 도착하여 최적화가 종료되기 전에 처리되지 않을 수 있습니다. OnTesterDeinit()에서 모든 늦은 프레임을 수신하려면 FrameNext() 함수를 사용하여 코드 블록을 배치하십시오.

더 보기

Testing trading strategies, Working with optimization results, TesterStatistics, OnTesterInit, OnTesterPass, ParameterGetRange, ParameterSetRange

© 2000-2025, MetaQuotes Ltd.

OnTesterPass

이 함수는 EA 최적화 중 새 데이터 프레임을 처리하기 위해 TesterPass 이벤트가 발생할 때 EA 단위로 호출됩니다.

```
void  OnTesterPass(void);
```

반환 값

반환값 없음

참고

TesterPass 이벤트는 Expert Advisor가 전략 테스터에서 최적화하는 동안 프레임을 받으면 자동으로 생성됩니다.

최적화 시작 중에 OnTesterDeInit() 또는 OnTesterPass() 이벤트 핸들러가 있는 EA가 별도의 터미널 차트에 자동으로 다운로드됩니다. 여기에는 테스터에 지정된 심볼과 기간이 있습니다. 이 함수는 최적화 중에 테스트 에이전트로부터 수신한 프레임을 처리하기 위한 것입니다. 테스트 결과가 포함된 프레임은 OnTester() 핸들러에서 FrameAdd() 함수를 사용하여 전송해야 합니다.

테스트 에이전트에서 FrameAdd() 함수를 사용하여 전송하는 최적화 프레임은 번들로 제공되며 제공에 시간이 걸릴 수 있습니다. 따라서 TesterPass 이벤트뿐 아니라 모든 프레임이 OnTesterPass()로 도착하여 최적화가 종료되기 전에 처리되지 않을 수 있습니다. OnTesterDeinit()에서 모든 늦은 프레임을 수신하려면 FrameNext() 함수를 사용하여 코드 블록을 배치하십시오.

After completing OnTesterDeinit() optimization, it is possible to sort all received frames again using the FrameFirst()/FrameFilter and FrameNext() functions.

더 보기

Testing trading strategies, Working with optimization results, OnTesterInit, OnTesterDeinit, FrameFirst, FrameFilter, FrameNext, FrameInputs

© 2000-2025, MetaQuotes Ltd.

마켓 정보 가져오기

이러한 기능은 시장 상태에 대한 정보를 수신하기 위한 기능입니다.

기능	액션
SymbolsTotal	사용 가능한 (Market Watch 또는 모두에서 선택된) 심볼 수를 반환
SymbolExist	Checks if a symbol with a specified name exists
SymbolName	지정된 심볼의 이름을 반환
SymbolSelect	마켓 워치(Market Watch) 창에서 기호를 선택하거나 창에서 심볼을 제거
SymbolIsSynchronized	터미널에서 선택한 심볼의 데이터가 거래 서버의 데이터와 동기화 되었는지 확인
SymbolInfoDouble	해당 속성에 대한 심볼 double 값을 반환
SymbolInfoInteger	해당 속성에 지정된 심볼에 대한 정수 유형(long, 날짜시간, int 또는 bool)의 값을 반환
SymbolInfoString	해당 속성에 대해 지정된 심볼에 대해 문자열 유형의 값을 반환
SymbolInfoMarginRate	주문 유형 및 방향에 따라 마진율을 반환
SymbolInfoTick	다음 유형의 변수에 지정된 심볼에 대한 현재 가격을 반환: MqlTick 유형
SymbolInfoSessionQuote	지정된 심볼 및 평일에 대해 지정된 시세 세션의 시작 및 종료 시간을 수신 허용.
SymbolInfoSessionTrade	지정된 심볼 및 평일에 대해 지정된 트레이딩 세션의 시작 및 종료 시간을 수신 허용.
MarketBookAdd	선택된 심볼에 대한 마켓의 깊이(Depth of Market) 열기를 제공하고 DOM 변경 알림을 수신하기 위해 구독.
MarketBookRelease	선택된 심볼에 대한 마켓의 깊이(Depth of Market) 닫기를 제공하고 DOM 변경 알림을 수신하기 위해 구독을 취소
MarketBookGet	지정된 기호의 마켓 뎁스 기록이 포함된 MqlBookInfo배열 구조를 반환

© 2000-2025, MetaQuotes Ltd.

SymbolsTotal

사용가능한 (선택된 Market Watch 또는 전체) 심볼의 수를 반환

```
int  SymbolsTotal(
   bool  selected      // True - MarketWatch의 유일한 심볼
   );
```

매개변수

선택됨

[in] 요청 코드. true 또는 false일 수 있음.

값 반환

만약 'selected' 변수가 참이면 이 함수는 MarketWatch에서 선택된 심볼의 수를 반환합니다. 만약 값이 false이면, 모든 심볼의 총 수를 반환

예:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용 가능한 심볼 수를 가져옵니다.
   int total    = SymbolsTotal(false);       // 서버에 있는 모든 심볼
   int selected = SymbolsTotal(true);        // 모든 심볼이 종합 시세창에 추가 되었습니다

//--- 얻은 데이터를 저널로 보냅니다.
   PrintFormat("Number of available symbols on the server: %d\n"+
               "Number of available symbols selected in MarketWatch: %d", total, selec
   /*
  결과:
   Number of available symbols on the server: 140
   Number of available symbols selected in MarketWatch: 11
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolExist

지정된 이름의 심볼이 있는지 확인.

```
bool  SymbolExist(
   const string  name,     // 심볼 이름
   bool&   is_custom       // 사용자 지정 심볼 속성
   );
```

매개변수

이름

[in] 심볼 이름.

is_custom

[out] 성공적으로 실행될 때 설정된 사용자 지정 심볼 속성. true일 경우, 탐지된 심볼은 사용자 지정 심볼입니다.

값 반환

false일 경우, 표준 및 사용자 지정심볼 중에서 찾을 수 없습니다.

예:

```
#define SYMBOL_NAME "GBPUSDn"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼 플래그를 선언하고 SYMBOL_NAME에 지정된 이름을 가진 심볼이 있는지 확인합니
   bool custom = false;
   bool result = SymbolExist(SYMBOL_NAME, custom);

//--- 디폴트로 '심볼를 찾을 수 없음' 메시지 텍스트 선언
   string text = StringFormat("The symbol '%s' was not found among either the standard

//--- 심볼이 발견되면 심볼이 발견된 목록에 따라 메시지 텍스트를 생성합니다.
   if(result)
     {
      //--- 만약 이것이 스탠다드 심볼이라면
      if(!custom)
         text = StringFormat("The '%s' symbol is available on the server.", SYMBOL_NAM
      //--- 만약 이것이 별도로 만든 심볼이라면
      else
         text = StringFormat("The symbol '%s' was found in the list of custom symbols.
     }

//--- 확인 결과와 관련한 메시지를 저널로 보냅니다.
   Print(text);
   /*
```

© 2000-2025, MetaQuotes Ltd.

```
   result for standard 'GBPUSD' symbol:
   The 'GBPUSD' symbol is available on the server.

   result for custom 'GBPUSDx' symbol:
   The symbol 'GBPUSDx' was found in the list of custom symbols.

   result for missing 'GBPUSDn' symbol:
   The symbol 'GBPUSDn' was not found among either the standard or custom symbols.
   */
  }
```

추가 참조

SymbolsTotal, SymbolSelect, Custom symbols

© 2000-2025, MetaQuotes Ltd.

SymbolName

심볼의 이름을 반환.

```
string  SymbolName(
   int   pos,           // 목록에 숫자를 기입
   bool  selected       // true - 오직 MarketWatch 내의 심볼인 경우
   );
```

매개변수

pos

[in] 심볼의 주문 번호.

선택됨

[in] 요청 코드. 값이 true이면 심볼은 MarketWatch에서 선택된 심볼 목록에서 불러져 옵니다. 값이 false이면 심볼이 일반 목록에서 제거됩니다.

값 반환

심볼 이름이 포함된 string 유형의 값.

예:

```
#define SYMBOL_NAME "GBPHKD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 서버 심볼 검색 결과의 플래그를 설정합니다.
   bool found = false;

//--- 모든 서버 심볼 목록에서 'SYMBOL_NAME' 심볼을 찾습니다.
   int total = SymbolsTotal(false);
   for(int i=0; i<total; i++)
     {
//--- 루프 인덱스로 목록에서 심볼명을 가져옵니다.
      string name = SymbolName(i, false);

//--- 이것이 원하는 심볼이라면 목록의 이름과 위치를 저널에 보내고 루프를 종료합니다.
      if(name == SYMBOL_NAME)
        {
         PrintFormat("The '%s' symbol was found in the list of server symbols at index
         found = true;
         break;
        }
     }

//--- 서버에서 심볼을 찾을 수 없으면 종료하기 전에 이를 보고합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!found)
      PrintFormat("The '%s' symbol was not found on the server.", SYMBOL_NAME);
}
```

© 2000-2025, MetaQuotes Ltd.

SymbolSelect

마켓 워치(Market Watch) 창에서 심볼을 선택하거나, 창에서 심볼을 제거합니다.

```
bool  SymbolSelect(
   string  name,        // 심볼 이름
   bool    select       // 추가 또는 제거
   );
```

매개변수

이름

[in] 심볼 이름.

선택

[in] 스위치. 값이 false이면 MarketWatch에서 심볼을 제거하고, 그렇지 않으면 이 창에서 심볼을 선택해야 합니다. 심볼 차트가 열려 있거나 이 심볼에 대한 열린 위치가 있는 경우 심볼을 제거할 수 없습니다.

값 반환

실패 시 false를 반환.

예:

```
#define SYMBOL_NAME "GBPHKD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 목록에 심볼이 있는지 확인하고 없으면 보고하고 작업을 완료합니다.
   bool custom = false;
   if(!SymbolExist(SYMBOL_NAME, custom))
     {
      PrintFormat("'%s' symbol not found in the lists", SYMBOL_NAME);
      return;
     }

//--- 종합시세 창에 심볼 추가
   ResetLastError();
   if(!SymbolSelect(SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }
//--- 목록에 심볼이 성공적으로 추가되면 종합시세 창에서 해당 심볼의 색인을 가져오고 결과를 저널에
   int index = SymbolIndex(SYMBOL_NAME);
   PrintFormat("The '%s' symbol has been added to the MarketWatch list. Symbol index
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 이제 종합시세 창에서 심볼을 제거합니다.
   ResetLastError();
   if(!SymbolSelect(SYMBOL_NAME, false))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }
//--- 심볼이 목록에서 성공적으로 제거되면 Market Watch 창의 해당 심볼 인덱스는 -1이고 삭제 결과
   index = SymbolIndex(SYMBOL_NAME);
   PrintFormat("The '%s' symbol has been removed from the MarketWatch list. Symbol inc

   /*
  결과:
   The 'GBPHKD' symbol has been added to the MarketWatch list. Symbol index in the lis
   The 'GBPHKD' symbol has been removed from the MarketWatch list. Symbol index in the
   */
  }
//+------------------------------------------------------------------+
//| 종합시세 심볼 목록의 심볼 인덱스 반환                                    |
//+------------------------------------------------------------------+
int SymbolIndex(const string symbol)
  {
   int total = SymbolsTotal(true);
   for(int i=0; i<total; i++)
     {
      string name = SymbolName(i, true);
      if(name == symbol)
         return i;
     }
   return(WRONG_VALUE);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolIsSynchronized

이 기능은 터미널에서 선택된 심볼과 데이터가 거래 서버의 데이터와 동기화되는지 여부를 확인.

```
bool  SymbolIsSynchronized(
   string  name,        // 심볼 이름
   );
```

매개변수

이름

[in] 심볼 이름.

값을 반환

데이터가 동기화되면 'true'를, 그렇지 않으면 'false'를 반환합니다.

예:

```
#define SYMBOL_NAME "EURUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 'SYMBOL_NAME' 심볼을 사용하여 터미널의 데이터를 서버 데이터와 동기화하기 위한 플래그를 가
   bool   sync = SymbolIsSynchronized(SYMBOL_NAME);

//--- 동기화 플래그에 따라 메시지를 생성합니다.
   string text = StringFormat("The data on the '%s' symbol in the terminal is synchron
   if(!sync)
      text = StringFormat("The data for the '%s' symbol in the terminal is not synchr

//--- 얻은 결과를 저널에 보냅니다.
   Print(text);

   /*
  동기화된 데이터의 결과:
   The data on the 'EURUSD' symbol in the terminal is synchronized with the data on th

  아직 동기화되지 않은 데이터에 대한 결과:
   The data for the 'GBPHKD' symbol in the terminal is not synchronized with the data
   */
  }
```

추가 참조

SymbolInfoInteger, Organizing Data Access

© 2000-2025, MetaQuotes Ltd.

SymbolInfoDouble

지정된 심볼에 해당하는 속성을 반환. 함수의 2가지 변수가 있습니다.

1. 속성 값을 즉시 반환.

```
double  SymbolInfoDouble(
   string                    name,        // 심볼
   ENUM_SYMBOL_INFO_DOUBLE   prop_id      // 속성 식별자
   );
```

2. 함수가 성공적으로 수행되었는지 여부에 따라 true 또는 false를 반환. 성공하는 경우 속성 값은 마지막 매개변수에서 참조하여 수신인 변수에 저장됩니다.

```
bool  SymbolInfoDouble(
   string                    name,        // 심볼
   ENUM_SYMBOL_INFO_DOUBLE   prop_id,     // 속성 식별자
   double&                   double_var   // 여기서 속성 값을 수락
   );
```

매개변수

이름

[in] 심볼 이름.

prop_id

[in] 심볼 속성 식별자. 값은 ENUM_SYMBOL_INFO_DOUBLE 열거 값 중 하나일 수 있습니다.

double_var

[out] 요청한 속성 값을 수신하는 double 유형의 변수.

값 반환

double 유형의 값. 실행에 실패한 경우 오류에 대한 정보는 다음 GetLastError() 기능을 사용하여 얻을 수 있습니다:

- 5040 – 심볼 이름을 지정하기 위한 잘못된 문자열 매개변수,
- 4301 – 알 수 없는 심볼(금융 상품),
- 4302 – "마켓 감시(Market Watch)"에서 심볼이 선택되지 않음(사용 가능한 심볼의 목록에서 찾을 수 없음),
- 4303 – 유효하지 않은 심볼 속성 식별자

주의

마지막 틱에 대한 정보를 가져오는 데 함수를 사용하는 경우 SymbolInfoTick()을 사용하는 것이 좋습니다. 터미널이 트레이딩 계좌에 연결되어 있기 때문에 아직 단 한 개의 시세도 나타나지 않았을 수 있습니다. 이 경우 요청된 값은 무한합니다.

대부분의 경우, 사용자가 한 번의 호출 동안 요청, 입찰, 마지막, 볼륨 및 마지막 틱 도착 시간 값을 받을 수 있도록 하는 SymbolInfoTick() 기능을 사용해도 충분합니다.

SymbolInfoMarginRate() 기능은 주문 유형 및 방향에 따라 청구된 마진의 양에 대한 데이터를 제공합니다.

예를 들어:

© 2000-2025, MetaQuotes Ltd.

```
void OnTick()
  {
//--- 심볼 속성에서 스프레드 얻기
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d points\r\n",
                            spreadfloat?"floating":"fixed",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- 이제 산포도를 스스로 계산하십시오
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Calculated spread = "+(string)spread_points+" points";
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoInteger

지정된 심볼에 해당하는 속성을 반환. 함수의 2가지 변수가 있습니다.

1. 속성 값을 즉시 반환.

```
long  SymbolInfoInteger(
   string                     name,      // 심볼
   ENUM_SYMBOL_INFO_INTEGER   prop_id    // 속성 식별자

   );
```

2. 함수가 성공적으로 수행되었는지 여부에 따라 true 또는 false를 반환. 성공하는 경우 속성 값은 마지막 매개변수에서 참조하여 수신인 변수에 저장됩니다.

```
bool  SymbolInfoInteger(
   string                     name,      // 심볼
   ENUM_SYMBOL_INFO_INTEGER   prop_id,   // 속성 식별자
   long&                      long_var   // 여기서 속성 값을 수락
   );
```

매개변수

이름

[in] 심볼 이름.

prop_id

[in] 심볼 속성 식별자. 값은 ENUM_SYMBOL_INFO_INTEGER 열거 값 중 하나일 수 있습니다.

long_var

[out] 요청된 속성 값을 수신하는 long 유형 변수.

값 반환

long 유형 값. 실행에 실패한 경우 오류에 대한 정보는 다음 GetLastError() 기능을 사용하여 얻을 수 있습니다:

- 5040 – 심볼 이름을 지정하기 위한 잘못된 문자열 매개변수,
- 4301 – 알 수 없는 심볼(금융 상품),
- 4302 – "마켓 감시(Market Watch)"에서 심볼이 선택되지 않음(사용 가능한 심볼의 목록에서 찾을 수 없음),
- 4303 – 유효하지 않은 심볼 속성 식별자

주의

마지막 틱에 대한 정보를 가져오는 데 함수를 사용하는 경우 SymbolInfoTick()을 사용하는 것이 좋습니다. 터미널이 트레이딩 계좌에 연결되어 있기 때문에 아직 단 한 개의 시세도 나타나지 않았을 수 있습니다. 이 경우 요청된 값은 무한합니다.

대부분의 경우, 사용자가 한 번의 호출 동안 요청, 입찰, 마지막, 볼륨 및 마지막 틱 도착 시간 값을 받을 수 있도록 하는 SymbolInfoTick() 기능을 사용해도 충분합니다.

예를 들어:

```
void OnTick()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 심볼 속성에서 스프레드 얻기
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d points\r\n",
                            spreadfloat?"floating":"fixed",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- 이제 산포도를 스스로 계산하십시오
   double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   double spread=ask-bid;
   int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
   comm=comm+"Calculated spread = "+(string)spread_points+" points";
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoString

지정된 심볼에 해당하는 속성을 반환. 함수의 2가지 변수가 있습니다.

1. 속성 값을 즉시 반환.

```
string  SymbolInfoString(
   string                   name,        // 심볼
   ENUM_SYMBOL_INFO_STRING  prop_id      // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 true 또는 false를 반환 성공하면 마지막 매개변수에서 참조로 전달된 자리표시자 변수에 속성 값이 배치됩니다.

```
bool  SymbolInfoString(
   string                   name,        // 심볼
   ENUM_SYMBOL_INFO_STRING  prop_id,     // 속성 식별자
   string&                  string_var   // 여기서 속성 값을 수락
   );
```

매개변수

이름

[in] 심볼 이름.

prop_id

[in] 심볼 속성 식별자. 값은 ENUM_SYMBOL_INFO_STRING 열거 값 중 하나일 수 있습니다.

string_var

[out] 요청된 속성 값을 수신하는 string 유형 변수.

값 반환

string 유형 값. 실행에 실패한 경우 오류에 대한 정보는 다음 GetLastError() 기능을 사용하여 얻을 수 있습니다:

- 5040 – 심볼 이름을 지정하기 위한 잘못된 문자열 매개변수,
- 4301 – 알 수 없는 심볼(금융 상품),
- 4302 – "마켓 감시(Market Watch)"에서 심볼이 선택되지 않음(사용 가능한 심볼의 목록에서 찾을 수 없음),
- 4303 – 유효하지 않은 심볼 속성 식별자

주의

마지막 틱에 대한 정보를 가져오는 데 함수를 사용하는 경우 SymbolInfoTick()을 사용하는 것이 좋습니다. 터미널이 트레이딩 계좌에 연결되어 있기 때문에 아직 단 한 개의 시세도 나타나지 않았을 수 있습니다. 이 경우 요청된 값은 무한합니다.

대부분의 경우, 사용자가 한 번의 호출 동안 요청, 입찰, 마지막, 볼륨 및 마지막 틱 도착 시간 값을 받을 수 있도록 하는 SymbolInfoTick() 기능을 사용해도 충분합니다.

예:

```
#define SYMBOL_NAME "USDJPY"
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- SYMBOL_NAME 심볼에 대한 문자열 데이터를 가져옵니다.
   string currency_base   = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_BASE);     //
   string currency_profit = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_PROFIT);   //
   string currency_margin = SymbolInfoString(SYMBOL_NAME, SYMBOL_CURRENCY_MARGIN);   //
   string symbol_descript = SymbolInfoString(SYMBOL_NAME, SYMBOL_DESCRIPTION);       //

//--- 획득한 데이터로 메시지 텍스트를 생성합니다.
   string text=StringFormat("Symbol %s:\n"+
                            "- Currency Base: %s\n"+
                            "- Currensy Profit: %s\n"+
                            "- Currency Margin: %s\n"+
                            "- Symbol Description: %s",
                            SYMBOL_NAME, currency_base,
                            currency_profit, currency_margin,
                            symbol_descript);

//--- 얻은 데이터를 저널로 보냅니다.
   Print(text);
   /*
   Symbol USDJPY:
   - Currency Base: USD
   - Currensy Profit: JPY
   - Currency Margin: USD
   - Symbol Description: US Dollar vs Yen
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoMarginRate

주문 유형 및 방향에 따라 마진율을 반환.

```
bool  SymbolInfoMarginRate(
   string             name,                    // 심볼 이름
   ENUM_ORDER_TYPE    order_type,              // 순서 유형
   double&            initial_margin_rate,     // 초기 마진율
   double&            maintenance_margin_rate  // 유지 마진율
   );
```

매개변수

이름

[in] 심볼 이름.

order_type

[in] 주문 유형.

initial_margin_rate

[in] 초기 마진율 수신하기 위한 double 유형 변수. 최초 마진은 적절한 방향의 1 lot 거래에 대한 보증금입니다. 최초 마진에 이율을 곱한 금액을 특정 유형의 주문 시 계좌에 적립해야 할 금액을 받습니다.

maintenance_margin_rate

[out] 유지 마진율을 수신하기 위한 double 유형 변수. 유지 마진은 1 lot의 개방 위치를 적절한 방향으로 유지하기 위한 최소 금액입니다. 유지 마진에 이율을 곱한 금액을 특정 유형의 주문이 활성화된 후 당좌에 적립해야 할 금액을 받습니다.

값 반환

속성이 대한 요청이 성공하는 경우 true를, 그렇지 않은 경우 false를 반환.

예:

```
#define SYMBOL_NAME Symbol()

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 비율이 설정된 변수를 선언합니다.
   double initial_margin_rate = 0;     // 초기 마진율
   double maintenance_margin_rate = 0; // 유지 마진율

//--- 매수 시장가 주문에 대한 SYMBOL_NAME 심볼 비율을 가져와서 저널로 보냅니다.
   SymbolInfoMarginRatePrint(SYMBOL_NAME, ORDER_TYPE_BUY, initial_margin_rate, mainter

//--- 매도 시장가 주문에 대한 SYMBOL_NAME 심볼 비율을 가져와서 저널로 보냅니다.
   SymbolInfoMarginRatePrint(SYMBOL_NAME, ORDER_TYPE_SELL, initial_margin_rate, mainte
```

© 2000-2025, MetaQuotes Ltd.

```
   /*
  결과:
   Initial margin rate for symbol AFKS for the Buy market order: 0.225600
   Maintenance margin rate for symbol AFKS for the Buy market order: 0.112800
   Initial margin rate for symbol AFKS for the Sell market order: 0.254400
   Maintenance margin rate for symbol AFKS for the Sell market order: 0.127200
   */
  }
//+------------------------------------------------------------------+
//| Send the margin ratios to the journal                            |
//+------------------------------------------------------------------+
void SymbolInfoMarginRatePrint(const string symbol, const ENUM_ORDER_TYPE order_type,
  {
//--- order_type 시장가 주문에 대한 '심볼' 비율을 가져옵니다.
   ResetLastError();
   if(!SymbolInfoMarginRate(symbol, order_type, initial_margin_rate, maintenance_margi
     {
      Print("SymbolInfoMarginRate() failed. Error ", GetLastError());
      return;
     }
//--- 얻은 비율 값을 인쇄합니다.
   string type=(order_type==ORDER_TYPE_BUY ? "Buy" : order_type==ORDER_TYPE_SELL ? "Se
   PrintFormat("Initial margin rate for symbol %s for the %s market order: %f\n"+
               "Maintenance margin rate for symbol %s for the %s market order: %f",
               SYMBOL_NAME, type, initial_margin_rate,
               SYMBOL_NAME, type, maintenance_margin_rate);
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoTick

함수는 MqlTick 유형의 변수에서 지정된 심볼의 현재 가격을 반환.

```
bool  SymbolInfoTick(
   string    symbol,     // 심볼 이름
   MqlTick&  tick        // 구조에 대한 참조
   );
```

매개변수

심볼

[in] 심볼 이름.

틱

[out] 마지막 가격 업데이트의 현재 가격 및 시간을 저장할 MqlTick 유형의 구조에 대한 링크.

값 반환

성공하면 함수가 true를 반환하고, 그렇지 않은 경우 false를 반환합니다.

예:

```
#define SYMBOL_NAME "EURUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 차원 1의 MqlTick 구조 유형을 사용하여 배열을 선언합니다.
   MqlTick tick[1]={};

//--- SYMBOL_NAME 심볼의 최신 가격을 MqlTick 구조로 가져옵니다.
   if(!SymbolInfoTick(SYMBOL_NAME, tick[0]))
     {
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return;
     }

//--- 얻은 데이터를 저널로 보냅니다.
   PrintFormat("Latest price data for the '%s' symbol:", SYMBOL_NAME);
   ArrayPrint(tick);
   /*
  결과:
   Latest price data for the 'EURUSD' symbol:
                    [time]    [bid]    [ask] [last] [volume]     [time_msc] [flags] [volu
   [0] 2024.05.17 23:58:54 1.08685 1.08695 0.0000         0 1715990334319       6
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

SymbolInfoSessionQuote

지정된 심볼 및 평일에 대해 지정된 시세 세션의 시작 및 종료 시간을 수신 허용.

```
bool  SymbolInfoSessionQuote(
   string            name,                // 심볼 이름
   ENUM_DAY_OF_WEEK  day_of_week,         // 요일
   uint              session_index,       // 세션 인덱스
   datetime&         from,                // 세션 시작 시간
   datetime&         to                   // 세션 종료 시간
   );
```

매개변수

이름

[in] 심볼 이름.

ENUM_DAY_OF_WEEK

[in] 요일, ENUM_DAY_OF_WEEK 열거 값.

uint

[in] 시작 시간과 종료 시간을 수신하려는 세션의 순서 번호. 세션 인덱싱은 0으로 시작.

from

[out] 반환된 값 날짜의 00시간 00분부터의 세션 시작 시간(초)은 무시해야 합니다.

to

[out] 반환된 값 날짜의 00시간 00분부터의 세션 종료 시간(초)은 무시해야 합니다.

값 반환

지정된 세션, 심볼 및 요일에 대한 데이터가 수신되면 true를, 그렇지 않은 경우 false를 반환합니다.

예:

```
#define SYMBOL_NAME    Symbol()
#define SESSION_INDEX 0

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 심볼과 SESSION_INDEX로 헤더를 인쇄하고
//--- 월요일부터 금요일까지 요일별 루프에서 시세 쿼트 세션의 시작 및 종료 시간을 저널에 인쇄합니다
   PrintFormat("Symbol %s, Quote session %d:", SYMBOL_NAME, SESSION_INDEX);
   for(int i=MONDAY; i<SATURDAY; i++)
      SymbolInfoSessionQuotePrint(SYMBOL_NAME, (ENUM_DAY_OF_WEEK)i, SESSION_INDEX);
   /*
  결과:
   Symbol RU000A103661, Quote session 0:
   - Monday     06:45 - 00:00
```

© 2000-2025, MetaQuotes Ltd.

```
   - Tuesday    06:45 - 00:00
   - Wednesday  06:45 - 00:00
   - Thursday   06:45 - 00:00
   - Friday     06:45 - 00:00
   */
  }
//+------------------------------------------------------------------+
//| 지정된 쿼트 세션의 시작 및 종료 시간 보내기                          |
//| 지정된 심볼 및 요일에 대한 저널                                     |
//+------------------------------------------------------------------+
void SymbolInfoSessionQuotePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- 쿼트 세션의 시작과 끝을 기록하는 변수를 선언합니다.
   datetime date_from;  // 세션 시작 시간
   datetime date_to;    // 세션 종료 시간

//--- 심볼 및 요일별로 쿼트 세션에서 데이터를 가져옵니다.
   if(!SymbolInfoSessionQuote(symbol, day_of_week, session_index, date_from, date_to))
     {
      Print("SymbolInfoSessionQuote() failed. Error ", GetLastError());
      return;
     }

//--- 열거형 상수로부터 요일 이름을 생성합니다.
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- 지정된 쿼트 세션에 대한 데이터를 저널로 보냅니다.
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

추가 참조

심볼 속성, TimeToStruct, 데이터 구조

© 2000-2025, MetaQuotes Ltd.

SymbolInfoSessionTrade

지정된 심볼 및 평일에 대해 지정된 트레이딩 세션의 시작 및 종료 시간을 수신 허용.

```
bool  SymbolInfoSessionTrade(
   string            name,                // 심볼 이름
   ENUM_DAY_OF_WEEK  day_of_week,         // 요일
   uint              session_index,       // 세션 인덱스
   datetime&         from,                // 세션 시작 시간
   datetime&         to                   // 세션 종료 시간
   );
```

매개변수

이름

[in] 심볼 이름.

ENUM_DAY_OF_WEEK

[in] 요일, ENUM_DAY_OF_WEEK 열거 값.

uint

[in] 시작 시간과 종료 시간을 수신하려는 세션의 순서 번호. 세션 인덱싱은 0으로 시작.

from

[out] 반환된 값 날짜의 00시간 00분부터의 세션 시작 시간(초)은 무시해야 합니다.

to

[out] 반환된 값 날짜의 00시간 00분부터의 세션 종료 시간(초)은 무시해야 합니다.

값을 반환

지정된 세션, 심볼 및 요일에 대한 데이터가 수신되면 true를, 그렇지 않은 경우 false를 반환합니다.

예:

```
#define SYMBOL_NAME    Symbol()
#define SESSION_INDEX 0

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 심볼과 SESSION_INDEX로 헤더를 인쇄하고
//--- 월요일부터 금요일까지 요일별 루프에서 거래 세션의 시작 및 종료 시간을 저널에 인쇄합니다.
   PrintFormat("Symbol %s, Trade session %d:", SYMBOL_NAME, SESSION_INDEX);
   for(int i=MONDAY; i<SATURDAY; i++)
      SymbolInfoSessionTradePrint(SYMBOL_NAME, (ENUM_DAY_OF_WEEK)i, SESSION_INDEX);
   /*
  결과:
   Symbol GBPUSD, Trade session 0:
   - Monday     00:15 - 23:55
```

© 2000-2025, MetaQuotes Ltd.

```
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   */
  }
//+------------------------------------------------------------------+
//| 지정된 거래 세션의 시작 및 종료 시간 보내기                          |
//| 지정된 심볼 및 요일에 대한 저널                                     |
//+------------------------------------------------------------------+
void SymbolInfoSessionTradePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- 거래 세션의 시작과 끝을 기록하는 변수를 선언합니다.
   datetime date_from;  // 세션 시작 시간
   datetime date_to;    // 세션 종료 시간

//--- 심볼 및 요일별로 거래 세션에서 데이터를 가져옵니다.
   if(!SymbolInfoSessionTrade(symbol, day_of_week, session_index, date_from, date_to))
     {
      Print("SymbolInfoSessionTrade() failed. Error ", GetLastError());
      return;
     }

//--- 열거형 상수로부터 요일 이름을 생성합니다.
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- 지정된 거래 세션에 대한 데이터를 저널로 보냅니다.
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

추가 참조

심볼 속성, TimeToStruct, 데이터 구조

© 2000-2025, MetaQuotes Ltd.

MarketBookAdd

선택된 심볼에 대한 마켓 깊이(Depth of Market)을 열고 DOM 변경 알림을 수신할 수 있는 구독을 제공합니다.

```
bool  MarketBookAdd(
   string  symbol      // 심볼
   );
```

매개변수

심볼

[in] 시장의 깊이가 엑스퍼트 어드바이저 또는 스크립트에 사용될 기호의 이름.

값 반환

성공적으로 열리면 true, 그렇지 않은 경우 false.

주의

일반적으로 이 기능은 OnInit() 함수 또는 클래스 생성기에서 호출해야 합니다. 들어오는 알림을 처리하려면 엑스퍼트 어드바이저 프로그램에서 OnBookEvent(string 및 symbol) 기능을 포함해야 합니다.

예:

```
#define   SYMBOL_NAME   "GBPUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- SYMBOL_NAME 심볼에 대한 시장 심도를 엽니다.
   if(!MarketBookAdd(SYMBOL_NAME))
     {
      PrintFormat("MarketBookAdd(%s) failed. Error ", SYMBOL_NAME, GetLastError());
      return;
     }

//--- 시장 심도를 성공적으로 열었다는 메시지를 저널에 보냅니다.
   PrintFormat("The MarketBook for the '%s' symbol was successfully opened and a subsc

//--- 2초 기다립니다
   Sleep(2000);

//--- 완료되면 공개 시장 심도에서 구독을 취소하세요.
   ResetLastError();
   if(MarketBookRelease(SYMBOL_NAME))
      PrintFormat("MarketBook for the '%s' symbol was successfully closed", SYMBOL_NAM
   else
      PrintFormat("Error %d occurred when closing MarketBook using the '%s' symbol", G
```

© 2000-2025, MetaQuotes Ltd.

```
 /*
결과:
 The MarketBook for the 'GBPUSD' symbol was successfully opened and a subscription t
 MarketBook for the 'GBPUSD' symbol was successfully closed
 */
}
```

추가 참조

시장 깊이의 구조, 구조 및 클래스

© 2000-2025, MetaQuotes Ltd.

MarketBookRelease

선택된 심볼에 대해 시장의 깊이(Depth of Market)를 마감하고 DOM 변경 알림 수신을 위한 구독을 취소.

```
bool  MarketBookRelease(
   string  symbol      // 심볼
   );
```

매개변수

심볼

[in] 심볼 이름.

값 반환

성공적으로 닫히는 경우 true 값을, 그렇지 않은 경우 false 값.

주의

일반적으로 OnDeinit() 기능에서 해당 MarketBookAdd() 기능이 호출된 경우, 이 기능은 OnInit() 기능에서 호출되어야 합니다. 또는 해당 MarketBookAdd() 기능을 클래스 생성자에서 호출한 경우 클래스 소멸기에서 호출해야 합니다.

예:

```
#define SYMBOL_NAME "GBPUSD"

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- SYMBOL_NAME 심볼에 대한 시장 심도를 엽니다.
   if(!MarketBookAdd(SYMBOL_NAME))
     {
      PrintFormat("MarketBookAdd(%s) failed. Error ", SYMBOL_NAME, GetLastError());
      return;
     }

//--- 시장 심도를 성공적으로 열었다는 메시지를 저널에 보냅니다.
   PrintFormat("The MarketBook for the '%s' symbol was successfully opened and a subsc

//--- 2초 기다립니다
   Sleep(2000);

//--- 완료되면 공개 시장 심도에서 구독을 취소하세요.
//--- 시장 심도에서 성공적으로 구독 취소되었거나 오류에 대한 메시지를 저널에 보냅니다.
   ResetLastError();
   if(MarketBookRelease(SYMBOL_NAME))
      PrintFormat("MarketBook for the '%s' symbol was successfully closed", SYMBOL_NAM
   else
      PrintFormat("Error %d occurred when closing MarketBook using the '%s' symbol", G
```

© 2000-2025, MetaQuotes Ltd.

```
 /*
결과:
 The MarketBook for the 'GBPUSD' symbol was successfully opened and a subscription t
 MarketBook for the 'GBPUSD' symbol was successfully closed
 */
}
```

추가 참조

마켓의 깊이의 구조, 구조 및 클래스

© 2000-2025, MetaQuotes Ltd.

MarketBookGet

지정된 심볼의 마켓의 깊이(Depth of Market) 기록을 포함하는 구조 배열 MqlBookInfo를 반환.

```
bool  MarketBookGet(
   string         symbol,      // 심볼
   MqlBookInfo&   book[]       // 배열에 대한 참조
   );
```

매개변수

심볼

[in] 심볼 이름.

book[]

[in] 마켓의 깊이(Depth of Market) 기록의 배열 참조. 충분한 수의 레코드를 위해 배열을 사전 할당할 수 있습니다. 동적 배열이 운영 메모리에 사전 할당되지 않은 경우 클라이언트 터미널이 배열 자체를 포함합니다.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

주의

MarketBookAdd() 기능을 통해 마켓의 깊이를 미리 열어야 합니다.

예를 들어:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo for ",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+":",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("DOM 심볼의 내용을 가져올 수 없습니다",Symbol());
     }
```

추가 참조

마켓의 깊이의 구조, 구조 및 클래스

© 2000-2025, MetaQuotes Ltd.

이코노믹 캘린더 기능

이 섹션에서 MetaTrader 플랫폼에서 직접 사용할 수 있는 이코노믹 캘린더로 작업하기 위한 기능에 대해 설명합니다. 이코노믹 캘린더는 거시경제 지표, 출시일, 중요도에 대한 설명이 수록된 미리 만들어진 백과사전입니다. 거시경제 지표의 관련 값은 발행 즉시 MetaTrader 플랫폼으로 전송되며 국가, 통화 및 중요도별로 필요한 지표를 시각적으로 추적할 수 있는 태그로 차트에 표시됩니다.

이코노믹 캘린더로 작업하는 모든 기능은 무역 서버 시간 (TimeTradeServer)을 사용합니다. 즉, MqlCalendarValue 구조 및 시간과 CalendarValueHistoryByEvent/CalendarValueHistory 기능의 시간 입력은 사용자의 로컬 시간이 아니라 거래 서버 시간대에 설정됩니다.

이코노믹 캘린더 기능은 필요한 국가/통화의 관점에서 맞춤형 중요도 기준에 따라 수신 이벤트의 자동 분석을 수행할 수 있도록 합니다.

기능	액션
CalendarCountryById	ID로 국가 설명을 가져오기
CalendarEventById	ID로 이벤트 설명을 가져오기
CalendarValueById	ID를 기준으로 이벤트 값 설명을 가져오기
CalendarCountries	캘린더에서 사용할 수 있는 국가명 배열을 가져오기
CalendarEventByCountry	캘린더에서 사용할 수 있는 모든 이벤트에 대한 설명 배열을 지정된 국가 코드로 가져오기
CalendarEventByCurrency	캘린더에서 사용할 수 있는 모든 이벤트에 대한 설명 배열을 지정된 통화로 가져오기
CalendarValueHistoryByEvent	지정된 시간 범위의 모든 이벤트에 대한 값 배열을 이벤트 ID를 기준으로 가져오기
CalendarValueHistory	지정된 시간 범위의 모든 이벤트에 대한 값의 배열을 정렬할 수 있는 국가 및 통화별로 가져오기
CalendarValueLastByEvent	change_id를 지정한 캘린더 데이터베이스 상태 이후 ID 별로 이벤트 값 배열을 가져오기
CalendarValueLast	change_id를 지정한 캘린더 데이터베이스 상태 이후 국가 및 통화별로 정렬할 수 있는 모든 이벤트의 값 배열을 가져오기

© 2000-2025, MetaQuotes Ltd.

CalendarCountryById

ID로 국가 설명 가져오기.

```
bool  CalendarCountryById(
   const long           country_id,     // 국가 ID
   MqlCalendarCountry&  country         // 국가 설명을 수신하기 위한 변수
   );
```

매개변수

country_id

[in] Country ID (ISO 3166-1).

country

[out] MqlCalendarCountry 국가 설명을 수신하기 위한 형식 변수.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻기 위해 GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 5402 – ERR_CALENDAR_NO_DATA (국가를 찾을 수 없음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과).

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 이코노믹 캘린더에서 국가 목록 가져오기
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- 결과 확인
   if(count==0)
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
//--- 만약 두 개 이상의 국가가 있다면
   if(count>=2)
     {
      MqlCalendarCountry country;
      //--- ID로 국가 설명을 가져오기
      if(CalendarCountryById(countries[1].id, country))
        {
         //--- 국가 설명 준비
         string descr="id = "+IntegerToString(country.id)+"\n";
         descr+=("name = " + country.name+"\n");
         descr+=("code = " + country.code+"\n");
         descr+=("currency = " + country.currency+"\n");
         descr+=("currency_symbol = " + country.currency_symbol+"\n");
```

© 2000-2025, MetaQuotes Ltd.

```
         descr+=("url_name = " + country.url_name);
         //--- 국가 설명 표시
         Print(descr);
        }
      else
         Print("CalendarCountryById() failed. Error ",GetLastError());
     }
//---
  }
/*
  결과:
   id = 999
   name = European Union
  코드 = EU
  통화 = EUR
   currency_symbol = €
   url_name = european-union
*/
```

추가 참조

CalendarCountries, CalendarEventByCountry

© 2000-2025, MetaQuotes Ltd.

CalendarEventById

ID로 이벤트 설명을 가져오기.

```
bool  CalendarEventById(
   ulong                event_id,      // 이벤트 ID
   MqlCalendarEvent&    event          // 이벤트 설명을 수신하기 위한 변수
   );
```

매개변수

event_id

[in] 이벤트 ID.

event

[out] 이벤트 설명을 수신하기 위한MqlCalendarEvent 유형 변수.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻기 위해 GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 5402 – ERR_CALENDAR_NO_DATA (국가를 찾을 수 없음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과).

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 독일의 국가 코드 (ISO 3166-1 Alpha-2)
   string germany_code="DE";
//--- 독일 이벤트 가져오기
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(germany_code,events);
//--- 저널에 독일 이벤트 표시
   if(events_count>0)
     {
      PrintFormat("독일의 이벤트: %d",events_count);
      ArrayPrint(events);
     }
   else
     {
      PrintFormat("국가 코드 %s, 오류 %d에 대한 이벤트를 수신하지 못했습니다.",
                  germany_code,GetLastError());
      //--- 스크립트 미리 완료
      return;
     }
//--- events[] 배열에서 마지막 이벤트에 대한 설명을 가져오기
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlCalendarEvent 이벤트;
   ulong event_id=events[events_count-1].id;
   if(CalendarEventById(event_id,event))
     {
      MqlCalendarCountry 국가;
      CalendarCountryById(event.country_id,country);
      PrintFormat("이벤트 설명과 event_id=%d가 수신됨",event_id);
      PrintFormat("국가: %s (country code = %d)",country.name,event.country_id);
      PrintFormat("이벤트 이름: %s",event.name);
      PrintFormat("이벤트 코드: %s",event.event_code);
      PrintFormat("이벤트 중요도: %s",EnumToString((ENUM_CALENDAR_EVENT_IMPORTANCE)even
      PrintFormat("이벤트 유형: %s",EnumToString((ENUM_CALENDAR_EVENT_TYPE)event.type))
      PrintFormat("이벤트 섹터: %s",EnumToString((ENUM_CALENDAR_EVENT_SECTOR)event.sect
      PrintFormat("이벤트 빈도: %s",EnumToString((ENUM_CALENDAR_EVENT_FREQUENCY)event.f
      PrintFormat("이벤트 해제 모드: %s",EnumToString((ENUM_CALENDAR_EVENT_TIMEMODE)ever
      PrintFormat("이벤트 측정 단위: %s",EnumToString((ENUM_CALENDAR_EVENT_UNIT)event.un
      PrintFormat("소수점 자릿 수: %d",event.digits);
      PrintFormat("이벤트 승수: %s",EnumToString((ENUM_CALENDAR_EVENT_MULTIPLIER)event.
      PrintFormat("소스 URL: %s",event.source_url);
     }
   else
      PrintFormat("event_d=%s, error %d에 대한 이벤트 설명을 수신하지 못했습니다",
                  event_id,GetLastError());
  }
/*
  결과:
  독일 이벤트: 50
              [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importa
   [ 0] 276010001      1        6           2           0          276      1
   [ 1] 276010002      1        6           2           0          276      1
   [ 2] 276010003      1        4           2           0          276      1
   [ 3] 276010004      1        4           2           0          276      1
   ....
   [47] 276500001      1        8           2           0          276      0
   [48] 276500002      1        8           2           0          276      0
   [49] 276500003      1        8           2           0          276      0
  event_id=276500003 가 수신된 이벤트 설명
  국가: 독 (country code = 276)
  이벤트명: Markit Composite PMI
  이벤트 코드: markit-composite-pmi
   이벤트 중요도: CALENDAR_IMPORTANCE_MODERATE
   이벤트 유형: CALENDAR_TYPE_INDICATOR
   이벤트 섹터: CALENDAR_SECTOR_BUSINESS
   이벤트 빈도: CALENDAR_FREQUENCY_MONTH
   이벤트 릴리스 모드: CALENDAR_TIMEMODE_DATETIME
   이벤트 측정 단위: CALENDAR_UNIT_NONE
  소수 자릿수: 1
   값 승수: CALENDAR_MULTIPLIER_NONE
   소스 URL: https://www.markiteconomics.com
```

© 2000-2025, MetaQuotes Ltd.

```
*/
```

추가 참조

CalendarEventByCountry, CalendarEventByCurrency, CalendarValueById

© 2000–2025, MetaQuotes Ltd.

CalendarValueById

its ID를 기준으로 이벤트 값 설명 가져오기.

```
bool  CalendarValueById(
   ulong                value_id,      // 이벤트 값 ID
   MqlCalendarValue&    value          // 이벤트 값을 수신하기 위한 변수
   );
```

매개변수

value_id

[in] 이벤트 값 ID.

값

[out] 이벤트 설명을 수신하기 위한 MqlCalendarValue 유형 변수. 다음을 확인 하세요캘린더 이벤트 핸들링 예제.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻기 위해 GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 5402 – ERR_CALENDAR_NO_DATA (국가를 찾을 수 없음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과).

주의

이코노믹 캘린더로 작업하는 모든 기능은 무역 서버 시간 (TimeTradeServer)을 사용합니다. 즉, MqlCalendarValue 구조 및 시간과 CalendarValueHistoryByEvent/CalendarValueHistory 기능의 시간 입력은 사용자의 로컬 시간이 아니라 거래 서버 시간대에 설정됩니다.

MqlCalendarValue structure는 actual_value, forecast_value, prev_value 와 revised_prev_value 필드의 value를 체크하고 세팅하는 메서드를 제공합니다. 만약 아무런 value도 없을 경우 필드는 LONG_MIN (-9223372036854775808)을 저장합니다.

이들 필드에 저장된 value에는 백만이 곱해집니다. 다음의 함수를 사용하여 MqlCalendarValu에서 밸류를 받을때 CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent and CalendarValueLast, 필드 밸류가 LONG_MIN과 같은지 확인해야 합니다; 만약 필드에 밸류가 지정된 경우 밸류를 얻기 위해서는 밸류를 1,000,000로 나누어 줘야 합니다. value를 가져오는 또 다른 방법은 MqlCalendarValue 구조의 함수를 사용하여 value를 확인하고 가져오는 것입니다.

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 일본의 국가 코드 (ISO 3166-1 Alpha-2)
   string japan_code="JP";
//--- 이벤트 발생 간격의 경계를 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   datetime date_from=D'01.01.2018';  // 2018년부터 모든 이벤트 받기
   datetime date_to=0;                // 0은 아직 발생하지 않은 이벤트를 포함하여 알려진 모든
//--- 일본 이벤트 값의 배열 가져오기
   MqlCalendarValue values[];
   int values_count=CalendarValueHistory(values,date_from,date_to,japan_code);
//--- 탐지된 이벤트 값을 따라 이동하기
   if(values_count>0)
     {
      PrintFormat("일본 이벤트 값의 수: %d",values_count);
      //--- 모든 "empty" 값 삭제 (actual_value==-9223372036854775808)
      for(int i=values_count-1;i>=0;i--)
        {
         if(values[i].actual_value==-9223372036854775808)
            ArrayRemove(values,i,1);
        }
      PrintFormat("빈 값 삭제 후의 값의 수: %d",ArraySize(values));
     }
   else
     {
      PrintFormat("국가 코드 %s, 오류 %d에 대한 이벤트를 수신하지 못했습니다.",
                  japan_code,GetLastError());
      //--- 스크립트 미리 완료
      return;
     }
//--- values[] 배열에 10개 이하의 값을 남기기
   if(ArraySize(values)>10)
     {
      PrintFormat("값 목록을 10으로 줄이고 표시");
      ArrayRemove(values,0,ArraySize(values)-10);
     }
   ArrayPrint(values);

//--- 이제 알려진 value_id를 기준으로 이벤트 값 설명을 가져오는 방법을 표시하겠습니다
   for(int i=0;i<ArraySize(values);i++)
     {
      MqlCalendarValue value;
      CalendarValueById(values[i].id,value);
      PrintFormat("%d: value_id=%d value=%d impact=%s",
                  i,values[i].id,value.actual_value,EnumToString(ENUM_CALENDAR_EVENT_I
     }
//---
  }
/*
  결과:
  일본 이벤트 값의 수: 1734
  빈 값 삭제 후 값의 수: 1017
  값의 목록을 10으로 줄이고 표시
        [id] [event_id]              [time]             [period] [revision] [actual_val
   [0] 56500  392030004 2019.03.28 23:30:00 2019.03.01 00:00:00          0          900
```

© 2000-2025, MetaQuotes Ltd.

```
   [1] 56501  392030005 2019.03.28 23:30:00 2019.03.01 00:00:00          0          700
   [2] 56502  392030006 2019.03.28 23:30:00 2019.03.01 00:00:00          0         1100
   [3] 56544  392030007 2019.03.28 23:30:00 2019.02.01 00:00:00          0         2300
   [4] 56556  392050002 2019.03.28 23:30:00 2019.02.01 00:00:00          0         1630
   [5] 55887  392020003 2019.03.28 23:50:00 2019.02.01 00:00:00          0          400
   [6] 55888  392020004 2019.03.28 23:50:00 2019.02.01 00:00:00          0        -1800
   [7] 55889  392020002 2019.03.28 23:50:00 2019.02.01 00:00:00          0          200
   [8] 55948  392020006 2019.03.28 23:50:00 2019.02.01 00:00:00          1         1400
   [9] 55949  392020007 2019.03.28 23:50:00 2019.02.01 00:00:00          1        -1000
  value_id를 기준으로 이벤트 값에 대한 간략한 데이터 표시
   0: value_id=56500 value=900000 impact=CALENDAR_IMPACT_POSITIVE
   1: value_id=56501 value=700000 impact=CALENDAR_IMPACT_NA
   2: value_id=56502 value=1100000 impact=CALENDAR_IMPACT_POSITIVE
   3: value_id=56544 value=2300000 impact=CALENDAR_IMPACT_NEGATIVE
   4: value_id=56556 value=1630000 impact=CALENDAR_IMPACT_POSITIVE
   5: value_id=55887 value=400000 impact=CALENDAR_IMPACT_NEGATIVE
   6: value_id=55888 value=-1800000 impact=CALENDAR_IMPACT_POSITIVE
   7: value_id=55889 value=200000 impact=CALENDAR_IMPACT_NEGATIVE
   8: value_id=55948 value=1400000 impact=CALENDAR_IMPACT_POSITIVE
   9: value_id=55949 value=-1000000 impact=CALENDAR_IMPACT_NEGATIVE
*/
```

추가 참조

CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent, CalendarValueLast

© 2000-2025, MetaQuotes Ltd.

CalendarCountries

캘린더에서 사용할 수 있는 국가명 배열을 가져오기

```
int  CalendarCountries(
   MqlCalendarCountry&  countries[]         // 캘린더 국가의 설명 목록을 수신하기 위한 배열
   );
```

매개변수

countries[]

[out] 모든 캘린더 국가의 설명을 수신하기 위한 MqlCalendarCountry 유형 배열.

값 반환

수신된 설명 수. 오류에 대한 정보를 얻기 위해GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과),
- 5400 – ERR_CALENDAR_MORE_DATA (배열 크기가 모든 국가의 설명을 수신하기에 불충분하며, 해당 국가의 설명만 수신됨).

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 이코노믹 캘린더에서 국가 목록 가져오기
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- 저널에서 배열 표시
   if(count>0)
      ArrayPrint(countries);
   else
      PrintFormat("CalendarCountries() returned 0! Error %d",GetLastError());
/*
  결과:
        [id]            [name] [code] [currency] [currency_symbol]        [url_name] [re
 [ 0]     0 "월드와이드"       "WW"   "ALL"      ""                 "월드와이드"
 [ 1]   999 "유럽 연합" "EU"   "EUR"      "€"                "유럽연합"           0
 [ 2]   840 "미국"   "US"   "USD"      "$"                "미국"             0
 [ 3]   124 "캐나다"          "CA"   "CAD"      "$"                "캐나다"
 [ 4]    36 "오스트레일리아"       "AU"   "AUD"      "$"                "오스트레일리아"
 [ 5]   554 "뉴질랜드"     "NZ"   "NZD"      "$"                "뉴질랜드"               0
 [ 6]   392 "일본"           "JP"   "JPY"      "¥"                "일본"
 [ 7]   156 "중국"           "CN"   "CNY"      "¥"                "중국"
 [ 8]   826 "영국" "GB"   "GBP"      "J"                "영국"           0
 [ 9]   756 "스위스"     "CH"   "CHF"      "₣"                "스위스"             0
 [10]   276 "독일"         "DE"   "EUR"      "€"                "독일"                 0
```

© 2000-2025, MetaQuotes Ltd.

```
   [11]  250 "프랑스"          "FR"   "EUR"      "€"                "프랑스"
   [12]  380 "이탈리아"          "IT"   "EUR"      "€"                "이탈리아"
   [13]  724 "스페인"           "ES"   "EUR"      "€"                "스페인"
   [14]   76 "브라질"           "BR"   "BRL"      "R$"               "브라질"
   [15]  410 "한국"     "KR"   "KRW"      "₩"              "한국"               0
*/
  }
```

추가 참조

CalendarEventByCountry, CalendarCountryById

© 2000-2025, MetaQuotes Ltd.

CalendarEventByCountry

캘린더에서 사용할 수 있는 모든 이벤트에 대한 설명 배열을 지정된 국가 코드로 가져오기

```
int  CalendarEventByCountry(
   string               country_code,     // 국가 코드명 (ISO 3166-1 alpha-2)
   MqlCalendarEvent&    events[]          // 설명 배열 수신을 위한 변수
   );
```

매개변수

country_code

[in] 국가 코드명 (ISO 3166-1 alpha-2)

events[]

[out] 지정된 국가에 대한 모든 이벤트에 대한 설명 수신을 위한MqlCalendarEvent 유형 배열.

값 반환

수신된 설명 수. 오류에 대한 정보를 얻기 위해GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 4004 – ERR_NOT_ENOUGH_MEMORY (요청된 작업을 위한 메모리가 충분하지 않음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과),
- 다음에 대한 실행 실패 오류: ArrayResize()

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- EU의 국가 코드 (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- EU 이벤트 가져오기
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- 저널에 EU 이벤트 표시
   if(events_count>0)
     {
      PrintFormat("EU events: %d",events_count);
      ArrayPrint(events);
     }
//---
  }
/*
  결과:
  EU 이벤트: 56
             [id] [type]  [country_id] [unit] [importance] [multiplier] [digits] [ever
  [ 0] 999010001      0           999      0            2            0        0  "ECB
  [ 1] 999010002      0           999      0            2            0        0  "ECB
```

© 2000-2025, MetaQuotes Ltd.

```
   [ 2] 999010003      0          999      0          3          0          0  "ECB
   [ 3] 999010004      0          999      0          3          0          0  "ECB
   [ 4] 999010005      0          999      0          2          0          0  "ECB
   [ 5] 999010006      1          999      1          3          0          2  "ECB
   [ 6] 999010007      1          999      1          3          0          2  "ECB
   [ 7] 999010008      0          999      0          2          0          0  "ECB
   [ 8] 999010009      1          999      2          2          3          3  "ECB
   [ 9] 999010010      0          999      0          2          0          0  "ECB
   [10] 999010011      0          999      0          2          0          0  "ECB
   ...

*/
```

추가 참조

CalendarCountries, CalendarCountryById

© 2000-2025, MetaQuotes Ltd.

CalendarEventByCurrency

캘린더에서 사용할 수 있는 모든 이벤트에 대한 설명 배열을 지정된 통화로 가져옵니다.

```
int  CalendarEventByCurrency(
   const string         currency,     // 국가 통화 코드명
   MqlCalendarEvent&    events[]      // 설명 배열을 받기 위한 변수
   );
```

매개변수

통화

[in] 국가 통화 코드명.

events[]

[out] 지정된 통화에 대한 모든 이벤트에 대한 설명 수신을 위한MqlCalendarEvent 유형 배열.

값 반환

수신된 설명 수. 오류에 대한 정보를 얻기 위해GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 4004 – ERR_NOT_ENOUGH_MEMORY (요청된 작업을 위한 메모리가 충분하지 않음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과),
- 다음에 대한 실행 실패 오류: ArrayResize()

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 이코노믹 캘린더 이벤트를 받을 준비를 선언
      MqlCalendarEvent events[];
//--- EU 통화 이벤트 가져오기
      int count = CalendarEventByCurrency("EUR",events);
      Print("count = ", count);
//--- 현재 예제를 위한 10개의 이벤트로 충분합니다
      if(count>10)
         ArrayResize(events,10);
//--- 저널에 이벤트 표시
      ArrayPrint(events);
  }
/*
  결과:
             [id] [type] [country_id] [unit] [importance]
   [0] 999010001      0          999      0            2 "https://www.ecb.europa.eu/ho
   [1] 999010002      0          999      0            2 "https://www.ecb.europa.eu/ho
   [2] 999010003      0          999      0            3 "https://www.ecb.europa.eu/ho
   [3] 999010004      0          999      0            3 "https://www.ecb.europa.eu/ho
```

© 2000-2025, MetaQuotes Ltd.

```
   [4] 999010005      0          999      0            2 "https://www.ecb.europa.eu/hc
   [5] 999010006      1          999      1            3 "https://www.ecb.europa.eu/hc
   [6] 999010007      1          999      1            3 "https://www.ecb.europa.eu/hc
   [7] 999010008      0          999      0            2 "https://www.ecb.europa.eu/hc
   [8] 999010009      1          999      2            2 "https://www.ecb.europa.eu/hc
   [9] 999010010      0          999      0            2 "https://www.ecb.europa.eu/hc
*/
```

추가 참조

CalendarEventById, CalendarEventByCountry

© 2000–2025, MetaQuotes Ltd.

CalendarValueHistoryByEvent

지정된 시간 범위의 모든 이벤트에 대한 값 배열을 이벤트 ID를 기준으로 가져오기

```
bool  CalendarValueHistoryByEvent(
   ulong              event_id,          // event ID
   MqlCalendarValue&  values[],          // 값 설명 배열
   datetime           datetime_from,     // 시간 범위의 좌측 테두리
   datetime           datetime_to=0      // 시간 범위의 우측 테두리
   );
```

매개변수

event_id

[in] 이벤트 ID.

values[]

[out] 이벤트 값을 수신하기 위한 MqlCalendarValue 유형 배열. 다음을 확인 하세요캘린더 이벤트 핸들링 예제.

datetime_from

[in] 시간 범위 이벤트의 초기 날짜는 지정된 아이디로 선택되고, 반면 *datetime_from* < *datetime_to*.

datetime_to=0

[in] 시간 범위 이벤트의 종료 날짜는 지정된 ID로 선택됩니다. *datetime_to* 가 설정되지 않은 경우 (또는 0인 경우), 캘린더 데이터베이스에서 지정된 *datetime_from*으로 시작하는 모든 이벤트 값이 반환됩니다 (향후 이벤트 값을 포함).

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻기 위해 GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 4004 – ERR_NOT_ENOUGH_MEMORY (요청된 작업을 위한 메모리가 충분하지 않음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과),
- 5400 – ERR_CALENDAR_MORE_DATA (배열 크기가 부족하여 모든 값에 대한 설명을 받을 수 없으며, 해당 값에 대한 설명만 수신됨),
- 다음에 대한 실행 실패 오류: ArrayResize()

주의

이코노믹 캘린더로 작업하는 모든 기능은 무역 서버 시간 (TimeTradeServer)을 사용합니다. 즉, MqlCalendarValue 구조 및 시간과 CalendarValueHistoryByEvent/CalendarValueHistory 기능의 시간 입력은 사용자의 로컬 시간이 아니라 거래 서버 시간대에 설정됩니다.

MqlCalendarValue structure는 actual_value, forecast_value, prev_value 와 revised_prev_value 필드의 value를 체크하고 세팅하는 메서드를 제공합니다. 만약 아무런 value도 없을 경우 필드는 LONG_MIN (-9223372036854775808)을 저장합니다.

이들 필드에 저장된 value에는 백만이 곱해집니다. 다음의 함수를 사용하여 MqlCalendarValu에서 밸류를 받을때 CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent and CalendarValueLast, 필드 밸류가 LONG_MIN과 같은지 확인해야 합니다;

© 2000-2025, MetaQuotes Ltd.

만약 필드에 밸류가 지정된 경우 밸류를 얻기 위해서는 밸류를 1,000,000로 나누어 줘야 합니다. value를 가져오는 또 다른 방법은 MqlCalendarValue 구조의 함수를 사용하여 value를 확인하고 가져오는 것입니다.

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- EU의 국가 코드 (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- EU 이벤트 가져오기
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- 저널에 EU 이벤트 표시
   if(events_count>0)
     {
      PrintFormat("EU events: %d",events_count);
      //--- 10개의 이벤트로 충분하니, 이벤트의 목록을 줄입니다
      ArrayResize(events,10);
      ArrayPrint(events);
     }
//--- "ECB 금리 결정" 이벤트에 event_id=999010007 가 있는지 확인합니다
   ulong event_id=events[6].id;        // 캘린더에서 이벤트 ID가 변경될 수 있으므로 반드시
   string event_name=events[6].name;   // 캘린더 이벤트의 이름
   PrintFormat("Get values for event_name=%s event_id=%d",event_name,event_id);
//--- "ECB 금리 결정" 이벤트의 모든 값을 가져오기
   MqlCalendarValue values[];
//--- 이벤트 발생 간격의 경계를 설정
   datetime date_from=0;           // 사용 가능한 내역의 시작부터 모든 내용을 받기
   datetime date_to=D'01.01.2016'; // 2016년 이전 이벤트를 받기
   if(CalendarValueHistoryByEvent(event_id,values,date_from,date_to))
     {
      PrintFormat("%s: %d에 대한 값을 수신하였습니다",
                  event_name,ArraySize(values));
      //--- 10개의 이벤트만 분석해도 충분하니, 값 목록을 줄입니다
      ArrayResize(values,10);
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Error! event_id=%d에 대한 값을 가져오지 못하였습니다",event_id);
      PrintFormat("Error code: %d",GetLastError());
     }
  }
//---
/*
```

© 2000-2025, MetaQuotes Ltd.

```
  결과:
  EU 이벤트: 56
            [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importar
   [0] 999010001      0        5           0           0          999      0
   [1] 999010002      0        5           0           0          999      0
   [2] 999010003      0        5           0           0          999      0
   [3] 999010004      0        5           0           0          999      0
   [4] 999010005      0        5           0           0          999      0
   [5] 999010006      1        5           0           0          999      1
   [6] 999010007      1        5           0           0          999      1
   [7] 999010008      0        5           0           0          999      0
   [8] 999010009      1        5           0           0          999      2
   [9] 999010010      0        5           0           0          999      0
  Get values for event_name=ECB Interest Rate Decision event_id=999010007
  수신된 ECB 금리 결정 이벤트 값: 102
       [id] [event_id]              [time]            [period] [revision] [actual_valu
   [0] 2776  999010007 2007.03.08 11:45:00 1970.01.01 00:00:00          0         3750(
   [1] 2777  999010007 2007.05.10 11:45:00 1970.01.01 00:00:00          0         3750(
   [2] 2778  999010007 2007.06.06 11:45:00 1970.01.01 00:00:00          0         4000(
   [3] 2779  999010007 2007.07.05 11:45:00 1970.01.01 00:00:00          0         4000(
   [4] 2780  999010007 2007.08.02 11:45:00 1970.01.01 00:00:00          0         4000(
   [5] 2781  999010007 2007.09.06 11:45:00 1970.01.01 00:00:00          0         4000(
   [6] 2782  999010007 2007.10.04 11:45:00 1970.01.01 00:00:00          0         4000(
   [7] 2783  999010007 2007.11.08 12:45:00 1970.01.01 00:00:00          0         4000(
   [8] 2784  999010007 2007.12.06 12:45:00 1970.01.01 00:00:00          0         4000(
   [9] 2785  999010007 2008.01.10 12:45:00 1970.01.01 00:00:00          0         4000(
*/
```

추가 참조

CalendarCountries, CalendarEventByCountry, CalendarValueHistory, CalendarEventById, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueHistory

국가 및 통화별로 정렬할 수 있는 기능을 사용하여 지정된 시간 범위의 모든 이벤트에 대한 값 배열을 가져오기.

```
bool  CalendarValueHistory(
   MqlCalendarValue&  values[],              // 값 설명을 위한 배열
   datetime           datetime_from,         // 시간 범위의 좌측 테두리
   datetime           datetime_to=0          // 시간 범위의 우측 테두리
   const string       country_code=NULL,     // 국가 코드명 (ISO 3166-1 alpha-2)
   const string       currency=NULL          // 국가 통화 코드명
   );
```

매개변수

values[]

[out] 이벤트 값을 수신하기 위한 MqlCalendarValue 유형 배열. 다음을 확인 하세요캘린더 이벤트 핸들링 예제.

datetime_from

[in] 시간 범위 이벤트의 초기 날짜는 지정된 아이디로 선택되고, 반면 *datetime_from* < *datetime_to*.

datetime_to=0

[in] 시간 범위 이벤트의 종료 날짜는 지정된 ID로 선택됩니다. *datetime_to* 가 설정되지 않은 경우 (또는 0인 경우), 캘린더 데이터베이스에서 지정된 *datetime_from*으로 시작하는 모든 이벤트 값이 반환됩니다 (향후 이벤트 값을 포함).

country_code=NULL

[in] 국가 코드명 (ISO 3166-1 alpha-2)

currency=NULL

[in] 국가 통화 코드명.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻기 위해 GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 4004 – ERR_NOT_ENOUGH_MEMORY (요청된 작업을 위한 메모리가 충분하지 않음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과),
- 5400 – ERR_CALENDAR_MORE_DATA (배열 크기가 부족하여 모든 값에 대한 설명을 받을 수 없으며, 해당 값에 대한 설명만 수신됨),
- 다음에 대한 실행 실패 오류: ArrayResize()

주의

이코노믹 캘린더로 작업하는 모든 기능은 무역 서버 시간 (TimeTradeServer)을 사용합니다. 즉, MqlCalendarValue 구조 및 시간과 CalendarValueHistoryByEvent/CalendarValueHistory 기능의 시간 입력은 사용자의 로컬 시간이 아니라 거래 서버 시간대에 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

고정된 길이의 *events[]* 배열이 함수에 전달되어 전체 결과를 저장할 공간이 충분하지 않으면 ERR_CALENDAR_MORE_DATA (5400) 오류가 활성화됩니다.

만약*datetime_to*가 설정되지 않은 경우(또는 0인 경우), 캘린더 데이터베이스에서 지정된 *datetime_from*으로 시작하는 모든 이벤트 값이 반환됩니다(향후 이벤트 값 포함).

country_code 및 *통화* 필터의 경우, NULL 및 "" 값은 동일하며 필터의 부재를 의미합니다.

*country_code*의 경우, *MqlCalendarCountry* 구조의코드필드(예:"US", "RU" 또는 "EU")를 사용해야 합니다.

*통화*의 경우, *MqlCalendarCountry* 구조의 통화 필드(예: "USD", "RUB" 또는 "EUR")를 사용해야 합니다.

필터는 연결로 적용되는데, 즉 논리적 'AND'는 두 조건(국가 및 통화)이 동시에 충족되는 이벤트 값만 선택하는데 사용됩니다.

MqlCalendarValue structure는 actual_value, forecast_value, prev_value 와 revised_prev_value 필드의 value를 체크하고 세팅하는 메서드를 제공합니다. 만약 아무런 value도 없을 경우 필드는 LONG_MIN (-9223372036854775808)을 저장합니다.

이들 필드에 저장된 value에는 백만이 곱해집니다. 다음의 함수를 사용하여 MqlCalendarValu에서 밸류를 받을때 CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent and CalendarValueLast, 필드 밸류가 LONG_MIN과 같은지 확인해야 합니다; 만약 필드에 밸류가 지정된 경우 밸류를 얻기 위해서는 밸류를 1,000,000로 나누어 줘야 합니다. value를 가져오는 또 다른 방법은 MqlCalendarValue 구조의 함수를 사용하여 value를 확인하고 가져오는 것입니다.

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- EU의 국가 코드 (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- 모든 EU 이벤트 값 가져오기
   MqlCalendarValue values[];
//--- 이벤트 발생 간격의 경계를 설정
   datetime date_from=D'01.01.2018';  // 2018년부터 모든 이벤트 받기
   datetime date_to=0;                // 0은 발생하지 않은 이벤트를 포함하여 알려진 모든 이
//--- 2018년 이후 EU 이벤트 내역 요청
   if(CalendarValueHistory(values,date_from,date_to,EU_code))
     {
      PrintFormat("country_code=%s: %d에 대한 이벤트 값을 수신하였습니다",
                  EU_code,ArraySize(values));
      //--- 저널로 출력할 배열의 크기를 줄입니다
      ArrayResize(values,10);
//--- 저널에 이벤트 값 표시
      ArrayPrint(values);
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("Error! country_code=%s",EU_code);에 대한 이벤트를 수신하지 못했습니다
      PrintFormat("Error code: %d",GetLastError());
     }
//---
  }
/*
  결과:
  country_code=EU: 1384에 대한 이벤트 값을 수신하였습니다
       [id] [event_id]           [time]               [period] [revision]    [actual_v
  [0] 54215  999500001 2018.01.02 09:00:00 2017.12.01 00:00:00           3        60600
  [1] 54221  999500002 2018.01.04 09:00:00 2017.12.01 00:00:00           3        56600
  [2] 54222  999500003 2018.01.04 09:00:00 2017.12.01 00:00:00           3        58100
  [3] 45123  999030005 2018.01.05 10:00:00 2017.11.01 00:00:00           0          600
  [4] 45124  999030006 2018.01.05 10:00:00 2017.11.01 00:00:00           0         2800
  [5] 45125  999030012 2018.01.05 10:00:00 2017.12.01 00:00:00           1          900
  [6] 45126  999030013 2018.01.05 10:00:00 2017.12.01 00:00:00           1         1400
  [7] 54953  999520001 2018.01.05 20:30:00 2018.01.02 00:00:00           0       127900
  [8] 22230  999040003 2018.01.08 10:00:00 2017.12.01 00:00:00           0         9100
  [9] 22231  999040004 2018.01.08 10:00:00 2017.12.01 00:00:00           0        18400
*/
```

추가 참조

CalendarCountries, CalendarEventByCountry, CalendarValueHistoryByEvent, CalendarEventById, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueLastByEvent

change_id가 지정된 캘린더 데이터베이스 상태 이후 ID별로 이벤트 값 배열을 가져오기.

```
int  CalendarValueLastByEvent(
   ulong                event_id,      // 이벤트 ID
   ulong&               change_id,     // 캘린더 변경 ID
   MqlCalendarValue&    values[]       // 값 설명을 위한 배열
   );
```

매개변수

event_id

[in] 이벤트 ID.

change_id

[in][out] ID 변경.

values[]

[out] 이벤트 값을 수신하기 위한 MqlCalendarValue 유형 배열. 다음을 확인 하세요캘린더 이벤트 핸들링 예제.

값 반환

수신된 이벤트 값의 수 오류에 대한 정보를 얻기 위해GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 4004 – ERR_NOT_ENOUGH_MEMORY (요청된 작업을 위한 메모리가 충분하지 않음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과),
- 5400 – ERR_CALENDAR_MORE_DATA (배열 크기가 부족하여 모든 값에 대한 설명을 받을 수 없으며, 해당 값에 대한 설명만 수신됨),
- 다음에 대한 실행 실패 오류: ArrayResize()

주의

이코노믹 캘린더로 작업하는 모든 기능은 무역 서버 시간 (TimeTradeServer)을 사용합니다. 즉, MqlCalendarValue 구조 및 시간과 CalendarValueHistoryByEvent/CalendarValueHistory 기능의 시간 입력은 사용자의 로컬 시간이 아니라 거래 서버 시간대에 설정됩니다.

고정된 길이의 ***events[]*** 배열이 함수에 전달되어 전체 결과를 저장할 공간이 충분하지 않으면 ERR_CALENDAR_MORE_DATA (5400) 오류가 활성화됩니다.

만약 *change_id* = 0이 함수에 전달되면 함수는 항상 0을 반환하지만, 현재 캘린더 데이터베이스는 *change_id*로 반환됩니다.

함수는 지정된 뉴스와 뉴스의 새로운 값을 수신하기 위해 함수의 후속 호출에 사용될 수 있는 새로운 *change_id*에 대한 배열을 반환합니다. 따라서, 마지막으로 알려진 *change_id*로 이 함수를 호출하여 지정된 뉴스에 대한 값을 업데이트할 수 있습니다.

MqlCalendarValue structure는 actual_value, forecast_value, prev_value 와 revised_prev_value 필드의 value를 체크하고 세팅하는 메서드를 제공합니다. 만약 아무런 value도 없을 경우 필드는 LONG_MIN (-9223372036854775808)을 저장합니다.

이들 필드에 저장된 value에는 백만이 곱해집니다. 다음의 함수를 사용하여 MqlCalendarValu에서 밸류를 받을때 CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory,

© 2000-2025, MetaQuotes Ltd.

CalendarValueLastByEvent and CalendarValueLast, 필드 밸류가 LONG_MIN과 같은지 확인해야 합니다; 만약 필드에 밸류가 지정된 경우 밸류를 얻기 위해서는 밸류를 1,000,000로 나누어 줘야 합니다. value를 가져오는 또 다른 방법은 MqlCalendarValue 구조의 함수를 사용하여 value를 확인하고 가져오는 것입니다.

Nonfarm 급여 보고서 릴리즈에 대한 EA 수신 대기 샘플:

```
#property 설명 "CalendarValueLastByEvent 함수 사용 예시"
#property 설명 " Nonfarm 급여 보고서의 공개를 추적용."
#property 설명 "이를 위해 현재 변경 ID 가져오기"
#property 설명 " 캘린더 데이터베이스의. 그 다음 수신하기 위해 이 ID를 사용"
#property 설명 " 타이머 조사를 통한 새로운 이벤트만"
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 함수                                             |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 타이머 생성
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 해제 함수                                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 타이머 파괴
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 함수                                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| 타이머 함수                                                      |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- 캘린더 데이터베이스 변경 ID
   static ulong calendar_change_id=0;
//--- 첫 번째 시작 속성
   static bool first=true;
//--- 이벤트 ID
   static ulong event_id=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 이벤트명
   static string event_name=NULL;
//--- 이벤트 값 배열
   MqlCalendarValue values[];
//--- 초기화 수행 - 현재 calendar_change_id 가져오기
   if(first)
     {
      MqlCalendarEvent events[];
      //--- 미국의 국가 코드 (ISO 3166-1 Alpha-2)
      string USA_code="US";
      //--- USA의 이벤트 가져오기
      int events_count=CalendarEventByCountry(USA_code,events);
      //--- '이벤트' 배열에서 필요한 이벤트의 포지션
      int event_pos=-1;
      //--- 저널에 USA 이벤트 표시
      if(events_count>0)
        {
         PrintFormat("%s: USA events: %d",__FUNCTION__,events_count);
         for(int i=0;i<events_count;i++)
           {
            string event_name_low=events[i].name;
            //--- 이벤트명을 소문자로 변경
            if(!StringToLower(event_name_low))
              {
               PrintFormat("StringToLower() returned %d error",GetLastError());
               //--- 시간 전에 나가기
               return;
              }
            //--- "논팜 페이롤" 이벤트를 찾기
            if(StringFind(event_name_low,"nonfarm payrolls")!=-1)
              {
               //--- 이벤트를 발견, 그 ID를 기억하기
               event_id=events[i].id;
               //--- "논팜 페이롤" 이벤트명 쓰기
               event_name=events[i].name;
               //--- 'events[]' 배열에 이벤트의 포지션 기억하기
               event_pos=i;
               //--- 캘린더 이름에 "논팜 페이롤"가 포함된 여러 이벤트가 있음을 명심하십시오
               PrintFormat("Event \"Nonfarm Payrolls\" found: event_id=%d  event_name=
               //--- 이 예시를 더 잘 이해하기 위해 '브레이크' 연산자를 언급하여 모든 이벤트를
               break;
              }
           }
         //--- "논팜 페이롤" 후 이벤트를 삭제하여 목록을 줄임
         ArrayRemove(events,event_pos+1);
         //--- 보다 편리한 분석을 위해 9개의 이벤트트를 "Nonfarm Payroll" 앞에 두기
         ArrayRemove(events,0,event_pos-9);
         ArrayPrint(events);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      else
        {
         PrintFormat("%s: CalendarEventByCountry(%s) returned 0 events, error code=%d"
                     USA_code,__FUNCTION__,GetLastError());
         //--- 작업이 실패로 완료되었으니 타이머의 다으 음 호출 중에 다시 시도하십시오
         return;
        }

      //--- 지정된 이벤트에 대한 캘린더 데이터베이스 변경 ID를 가져오기
      if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
        {
         //--- 이 코드 블록은 첫 번째 실행 중에 실행할 수 없지만 그래도 추가해 보십시오
         PrintFormat(""%s: Received the Calendar database current ID: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- 타이머의 다음 이벤트 전에 플래그를 설정하고 종료하십시오
         first=false;
         return;
        }
      else
        {
         //--- 데이터가 수신되지 않았으니(첫 번째 실행에서는 정상), 오류를 확인하십시오
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: 캘린더 데이터베이스 현재 ID를 수신했습니다: change_id=%d",
                        __FUNCTION__,calendar_change_id);
            //--- 타이머의 다음 이벤트 전에 플래그를 설정하고 종료십시오
            first=false;
            //--- 이제 calendar_change_id 값을 가집니다
            return;
           }
         else
           {
            //--- 그리고 이는 진짜 오류입니다
            PrintFormat("%s: event_id=%d에 대한 값을 가져오는 데 실패했습니다",__FUNCTION_
            PrintFormat("Error code: %d",error_code);
            //--- 작업이 실패로 완료되었으니 타이머의 다음 호출 중에 다시 시도하십시오
            return;
           }
        }
     }

//--- 캘린더 변경 ID (change_id)의 마지막 알려진 값이 있습니다.
   ulong old_change_id=calendar_change_id;
//--- Nonfarm Payrolls 이벤트 값을 확인
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
     {
      PrintFormat("%s: 새 이벤트를 수신 ₩"%s\": %d",
                  __FUNCTION__,event_name,ArraySize(values));
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 저널에 'values' 배열의 데이터를 표시
      ArrayPrint(values);
      //--- 저널에 이전 및 새 캘린더 ID의 값을 표시
      PrintFormat("%s: 이전 change_id=%d, new change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
/*
      여기에 "논팜 페이롤" 데이터 릴리스를 처리하는 코드를 쓰기
      */
     }
//---
  }
/*
  결과:
   OnTimer: USA events: 202
  Event "Nonfarm Payrolls" found: event_id=840030016  event_name=Nonfarm Payrolls
           [id] [type] [sector] [frequency] [time_mode] [country_id] [unit] [importar
   [0] 840030007      1        4           2           0           840       1
   [1] 840030008      1        4           2           0           840       1
   [2] 840030009      1        4           2           0           840       0
   [3] 840030010      1        4           2           0           840       0
   [4] 840030011      1        4           2           0           840       1
   [5] 840030012      1        4           2           0           840       1
   [6] 840030013      1        4           2           0           840       1
   [7] 840030014      1        4           2           0           840       1
   [8] 840030015      1        3           2           0           840       1
   [9] 840030016      1        3           2           0           840       4
   OnTimer: 캘린더 데이터베이스 현재 ID를 수신했습니다: change_id=33986560

*/
```

추가 참조

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

CalendarValueLast

change_id를 지정한 캘린더 데이터베이스 상태 이후 국가 및 통화별로 정렬할 수 있는 모든 이벤트의 값 배열을 가져오기.

```
int  CalendarValueLast(
   ulong&               change_id,             // ID 변경
   MqlCalendarValue&    values[],              // 값 설명을 위한 배열
   const string         country_code=NULL,     // 국가 코드명 (ISO 3166-1 alpha-2)
   const string         currency=NULL          // 국가 통화 코드명
   );
```

매개변수

change_id

[in][out] ID 변경.

values[]

[out] 이벤트 값을 수신하기 위한 MqlCalendarValue 유형 배열. 다음을 확인 하세요캘린더 이벤트 핸들링 예제.

country_code=NULL

[in] 국가 코드명 (ISO 3166-1 alpha-2)

currency=NULL

[in] 국가 통화 코드명.

값 반환

수신된 이벤트 값의 수 오류에 대한 정보를 얻기 위해GetLastError() 함수 호출. 가능한 오류:

- 4001 – ERR_INTERNAL_ERROR (일반 실행시간 오류),
- 4004 – ERR_NOT_ENOUGH_MEMORY (요청된 작업을 위한 메모리가 충분하지 않음),
- 5401 – ERR_CALENDAR_TIMEOUT (요청시간 제한 초과),
- 5400 – ERR_CALENDAR_MORE_DATA (배열 크기가 부족하여 모든 값에 대한 설명을 받을 수 없으며, 해당 값에 대한 설명만 수신됨),
- 다음에 대한 실행 실패 오류: ArrayResize()

주의

이코노믹 캘린더로 작업하는 모든 기능은 무역 서버 시간 (TimeTradeServer)을 사용합니다. 즉, MqlCalendarValue 구조 및 시간과 CalendarValueHistoryByEvent/CalendarValueHistory 기능의 시간 입력은 사용자의 로컬 시간이 아니라 거래 서버 시간대에 설정됩니다.

고정된 길이의 *events[]* 배열이 함수에 전달되어 전체 결과를 저장할 공간이 충분하지 않으면 ERR_CALENDAR_MORE_DATA (5400) 오류가 활성화됩니다.

change_id = 0이 함수에 전달되면 해당 매개 변수에 대한 캘린더 데이터베이스의 현재 *change_id*가 표시되고 함수는 0을 반환합니다

country_code 및 *통화* 필터의 경우, NULL 및 "" 값은 동일하며 필터의 부재를 의미합니다.

*country_code*의 경우, *MqlCalendarCountry* 구조의코드필드(예:"US", "RU" 또는 "EU")를 사용해야 합니다.

© 2000-2025, MetaQuotes Ltd.

*통화*의 경우, *MqlCalendarCountry* 구조의 통화 필드(예: "USD", "RUB" 또는 "EUR")를 사용해야 합니다.

필터는 연결로 적용되는데, 즉 논리적 'AND'는 두 조건(국가 및 통화)이 동시에 충족되는 이벤트 값만 선택하는데 사용됩니다

함수는 지정된 뉴스와 뉴스의 새로운 값을 수신하기 위해 함수의 후속 호출에 사용될 수 있는 새로운 *change_id*에 대한 배열을 반환합니다. 따라서, 마지막으로 알려진 *change_id*로 이 함수를 호출하여 지정된 뉴스에 대한 값을 업데이트할 수 있습니다.

MqlCalendarValue structure는 actual_value, forecast_value, prev_value 와 revised_prev_value 필드의 value를 체크하고 세팅하는 메서드를 제공합니다. 만약 아무런 value도 없을 경우 필드는 LONG_MIN (-9223372036854775808)을 저장합니다.

이들 필드에 저장된 value에는 백만이 곱해집니다. 다음의 함수를 사용하여 MqlCalendarValu에서 밸류를 받을때 CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent and CalendarValueLast, 필드 밸류가 LONG_MIN과 같은지 확인해야 합니다; 만약 필드에 밸류가 지정된 경우 밸류를 얻기 위해서는 밸류를 1,000,000로 나누어 줘야 합니다. value를 가져오는 또 다른 방법은 MqlCalendarValue 구조의 함수를 사용하여 value를 확인하고 가져오는 것입니다.

이코노믹 캘린더 이벤트에 대한 EA 수신 대기 샘플:

```
#property 설명 "CalendarValueLast 함수 사용 예제"
#property 설명 " 이코노믹 캘린더 이벤트를 청취하는 사람을 개발"
#property 설명 "이를 위해 현재 변경 ID 가져오기"
#property 설명 " 캘린더 데이터베이스의. 그 다음 수신하기 위해 이 ID를 사용"
#property 설명 " 타이머 조사를 통한 새로운 이벤트만"
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 함수                                             |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 타이머 생성
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 해제 함수                                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 타이머 파괴
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 함수                                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 타이머 함수                                                      |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- 캘린더 데이터베이스 변경 ID
   static ulong calendar_change_id=0;
//--- 첫 번째 시작 속성
   static bool first=true;
//--- 이벤트 값 배열
   MqlCalendarValue values[];
//--- 초기화 수행 - 현재 calendar_change_id 가져오기
   if(first)
     {
      //--- 일정관리 데이터베이스 변경 ID를 가져오기
      if(CalendarValueLast(calendar_change_id,values)>0)
        {
         //--- 이 코드 블록은 첫 번째 실행 중에 실행할 수 없지만 그래도 추가해 보십시오
         PrintFormat(""%s: Received the Calendar database current ID: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- 타이머의 다음 이벤트 전에 플래그를 설정하고 종료하십시오
         first=false;
         return;
        }
      else
        {
         //--- 데이터가 수신되지 않았으니(첫 번째 실행에서는 정상), 오류를 확인하십시오
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: 캘린더 데이터베이스 현재 ID를 수신했습니다: change_id=%d",
                        __FUNCTION__,calendar_change_id);
            //--- 타이머의 다음 이벤트 전에 플래그를 설정하고 종료십시오
            first=false;
            //--- 이제 calendar_change_id 값을 가집니다
            return;
           }
         else
           {
            //--- 그리고 이는 진짜 오류입니다
            PrintFormat("%s: CalendarValueLast에서 이벤트를 가져오는 데 실패했습니다. 오류
                        __FUNCTION__,error_code);
            //--- 작업이 실패로 완료되었으니 타이머의 다음 호출 중에 다시 초기화
            return;
           }
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 캘린더 변경 ID (change_id)의 마지막 알려진 값이 있습니다.
   ulong old_change_id=calendar_change_id;
//--- 새 캘린더 이벤트가 있는지 확인
   if(CalendarValueLast(calendar_change_id,values)>0)
     {
      PrintFormat("%s: 새 캘린더 이벤트 수신됨: %d",
                  __FUNCTION__,ArraySize(values));
      //--- 저널에 'values' 배열의 데이터를 표시
      ArrayPrint(values);
      //--- 저널에 이전 및 새 캘린더 ID의 값을 표시
      PrintFormat("%s: 이전 change_id=%d, new change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
      //--- 저널에 새 이벤트 표시
      ArrayPrint(values);
      /*
     여기에 이벤트 발생을 처리하는 코드를 쓰십시오
      */
     }
//---
  }
/*
  수신기 작업 예시:
  OnTimer: 캘린더 데이터베이스 현재 ID를 수신했습니다: change_id=33281792
  OnTimer: 캘린더에 대한 새 이벤트를 수신했습니다: 1
        [id] [event_id]             [time]            [period] [revision] [actual_va
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -507
  OnTimer: 이전 change_id=33281792, 새 change_id=33282048
        [id] [event_id]             [time]            [period] [revision] [actual_va
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -507
  OnTimer: 캘린더에 대한 새 이벤트를 수신했습니다: 1
        [id] [event_id]             [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203
  OnTimer: 이전 change_id=33282048, 새 change_id=33282560
        [id] [event_id]             [time]            [period] [revision]       [actu
   [0] 91041   76020013 2019.03.27 15:30:00 1970.01.01 00:00:00          0 -922337203

*/
```

추가 참조

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

© 2000-2025, MetaQuotes Ltd.

시계열 및 지표 데이터 액세스

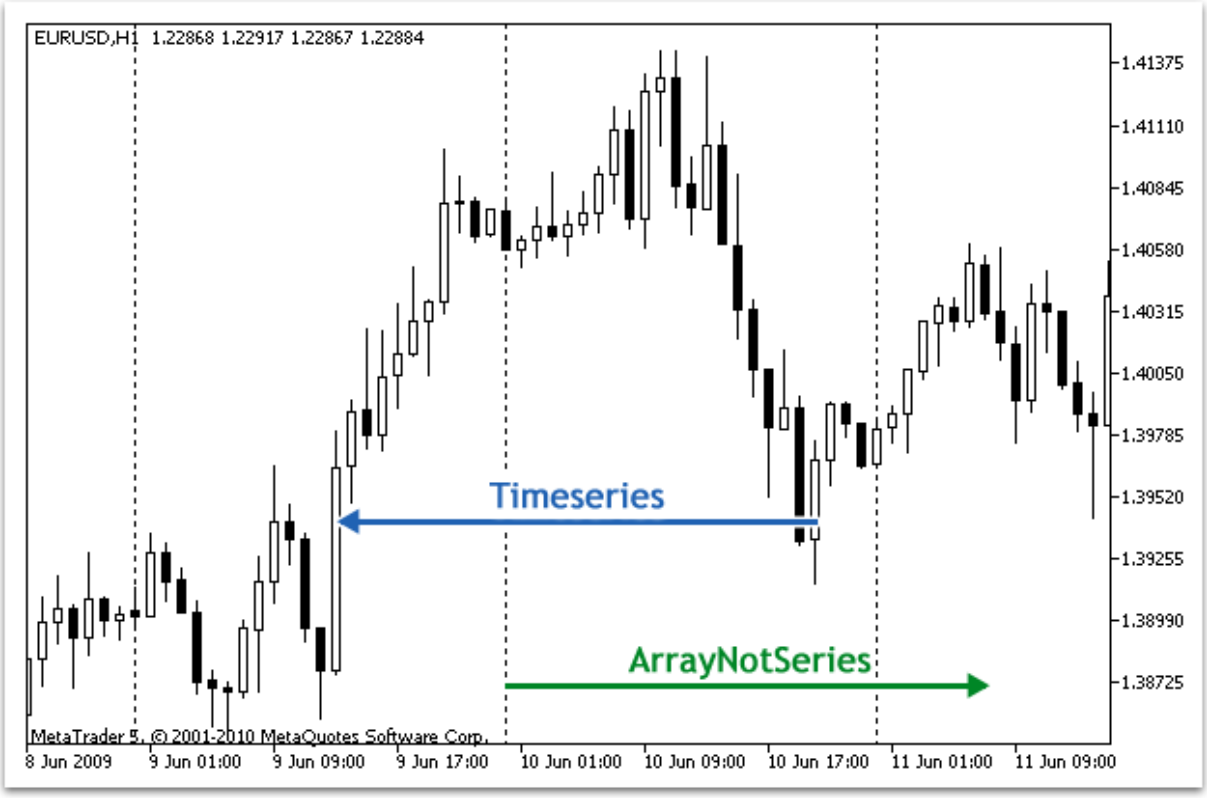
이러한 함수는 시계열 및 지표로 작업하기 위한 것입니다. 시계열은 역순으로 일반적인 데이터 배열과 다릅니다. 시계열의 요소는 배열 끝에서 시작(가장 최근의 데이터부터 가장 오래된 데이터까지)까지 색인화됩니다. 복제 함수는 값을 수신하는 배열의 필요한 크기를 할당하도록 설계되었기 때문에 시계열 값과 지시자 데이터를 복사하려면 동적 배열만 사용하는 것이 좋습니다.

이 규칙에는 **중요한 예외** 가 있습니다: 예를 들어, Expert Advisor에서 OnTick()의 각 호출 또는 지표에서 OnCalculate()의 각 호출 시 시계열 및 지시자 값을 자주 복사해야 하는 경우, 이 때 정적 분산 배열을 더 잘 사용해야 하는데, **메모리 할당 작업은** 동적 배열**에 대해선 추가 시간이 필요하며**, 이는 테스트 및 최적화 중에 영향을 미치기 때문입니다.

시계열 및 지표 값에 액세스하는 함수를 사용할 경우 색인화 방향을 고려해야 합니다. 이 내용은 배열, 버퍼 및 시계열의 인덱싱 방향 섹션에서 설명됩니다.

지표 및 시계열 데이터에 대한 액세스는 요청된 데이터가 준비되었는지 여부에 관계없이 구현됩니다(일명 비동기 액세스). 이는 커스텀 지표 계산에 매우 중요하므로 데이터가 없으면 *Copy...()* 유형의 함수가 즉시 오류를 반환합니다. 그러나 Expert Advisors 및 스크립트에 액세스할 때 필요한 시계열을 다운로드하거나 지표 값을 계산하는 데 필요한 시간을 제공하기 위해 데이터를 수신하는 몇 번의 시도가 약간씩 일시 중지됩니다.

데이터 액세스 구조화 섹션에서는 MetaTrader 5 클라이언트 터미널에서 가격 데이터의 수신, 저장 및 요청에 대한 세부 정보를 설명합니다.



배열의 가격 데이터에 대한 액세스는 데이터 끝에서 수행된다는 것은 역사상 인정됩니다. 물리적으로 새 데이터는 항상 어레이 끝에 기록되지만 어레이의 인덱스는 항상 0과 같습니다. 시계열 배열의 0 인덱스는 현재 막대의 데이터, 즉 이 시간대의 미완료 시간 간격에 해당하는 막대를 나타냅니다.

© 2000-2025, MetaQuotes Ltd.

기간(타임프레임)은 단일 가격 막대가 형성되는 기간을 뜻합니다. 21개의 미리 정의된 표준 기간(타임프레임)이 있습니다.

함수	기능
SeriesInfoInteger	내역 데이터의 상태에 대한 정보를 반환합니다
Bars	지정된 심볼 및 기간에 대한 내역의 막대 수를 반환합니다
BarsCalculated	오류 발생 시 지표 버퍼 또는 -1의 계산된 데이터 수를 반환합니다(데이터가 아직 계산되지 않음).
IndicatorCreate	MqlParam 유형 매개 변수 배열에서 생성된 지정된 기술 지표로 핸들을 반환합니다.
IndicatorParameters	지정된 핸들을 기준으로, 지표의 입력 매개 변수 수와 매개 변수의 값 및 유형을 반환합니다.
IndicatorRelease	다른 사용자가 사용하지 않는 경우 지표 핸들을 제거하고 지표의 계산 블록을 해제합니다.
CopyBuffer	지정된 지표에서 배열로 지정된 버퍼의 데이터를 가져옵니다.
CopyRates	지정된 심볼 및 기간에 대한 요율 구조의 내역 데이터를 배열로 가져옵니다
CopySeries	지정된 심볼-기간 및 지정된 양만큼 MqlRates 구조에서 동기화된 시계열을 가져옵니다. 데이터는 지정된 배열 세트로 수신됩니다
CopyTime	지정된 심볼 및 기간에 대한 막대 오프닝 시간의 내역 데이터를 배열로 가져옵니다
CopyOpen	지정된 심볼 및 기간에 대한 막대 시가에 대한 내역 데이터를 배열로 가져옵니다
CopyHigh	지정된 심볼 및 기간에 대한 막대 최대 가격에 대한 내역 데이터를 배열로 가져옵니다
CopyLow	지정된 심볼 및 기간에 대한 막대 최소 가격에 대한 내역 데이터를 배열로 가져옵니다
CopyClose	지정된 심볼 및 기간에 대한 종가 막대에 대한 내역 데이터를 배열로 가져옵니다.
CopyTickVolume	지정된 심볼 및 기간에 대한 틱의 볼륨의 내역 데이터를 배열로 가져옵니다.
CopyRealVolume	지정된 심볼 및 기간의 거래 볼륨에 대한 내역 데이터를 어레이로 가져옵니다.
CopySpread	지정한 심볼 및 기간에 대한 스프레드시트의 내역 데이터를 배열로 가져옵니다.
CopyTicks	MqlTick 형식의 틱을 ticks_array로 가져옵니다
CopyTicksRange	지정된 날짜 범위 내의 MqlTick 형식의 틱을 ticks_array로 가져옵니다
iBars	내역에서 사용할 수 있는 해당 심볼 및 마침표의 막대 수를 반환합니다.
iBarShift	지정된 시간에 해당하는 막대의 인덱스를 반환합니다

© 2000-2025, MetaQuotes Ltd.

함수	기능
iClose	해당 차트에서 막대의 종가('shift' 매개 변수로 표시됨)를 반환합니다
iHigh	해당 차트에서 막대의 고가('Shift' 매개 변수로 표시됨)를 반환합니다
iHighest	해당 차트에 있는 가장 높은 값의 인덱스를 반환합니다(현재 막대에 상대적인 이동)
iLow	해당 차트에서 바의 저가('Shift' 매개 변수로 표시됨)를 반환합니다
iLowest	해당 차트에 있는 최소 값의 인덱스를 반환합니다(현재 막대에 상대적인 이동)
iOpen	해당 차트에서 막대의 시가('Shift' 매개 변수로 표시됨)를 반환합니다
iTime	해당 차트에서 막대의 시작 시간('Shift' 파라미터로 표시됨)을 반환합니다
iTickVolume	해당 차트에서 막대의 틱 볼륨('Shift' 파라미터로 표시됨)을 반환합니다
iRealVolume	해당 차트에서 막대의 실제 볼륨('Shift' 파라미터로 표시됨)을 반환합니다
iVolume	해당 차트에서 막대의 틱 볼륨('Shift' 파라미터로 표시됨)을 반환합니다
iSpread	해당 차트에서 막대의 스프레드 값('Shift' 매개 변수로 표시됨)을 반환합니다

ArraySetAsSeries() 함수를 사용하면 시계열에서와 같은 요소에 대한 배열 액세스를 설정할 수 있지만, 어레이 요소는 동일한 순서로 물리적으로 저장되어 인덱싱 방향만 변경됩니다. 이 사실을 입증하기 위해 예를 들어 보겠습니다:

```
   datetime TimeAsSeries[];
//--- 시계열처럼 배열에 대한 액세스를 설정합니다
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
   if(copied<=0)
     {
      Print("최근 10개의 막대에 대한 시가의 값의 복제 작업이 실패했습니다.");
      return;
     }
   Print("TimeCurrent =",TimeCurrent());
   Print("ArraySize(Time) =",ArraySize(TimeAsSeries));
   int size=ArraySize(TimeAsSeries);
   for(int i=0;i<size;i++)
     {
      Print("TimeAsSeries["+i+"] =",TimeAsSeries[i]);
     }

   datetime ArrayNotSeries[];
   ArraySetAsSeries(ArrayNotSeries,false);
   ResetLastError();
   copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
   if(copied<=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("최근 10개의 막대에 대한 시가의 값의 복제 작업이 실패했습니다.");
      return;
     }
   size=ArraySize(ArrayNotSeries);
   for(int i=size-1;i>=0;i--)
     {
      Print("ArrayNotSeries["+i+"] =",ArrayNotSeries[i]);
     }
```

그 결과 다음과 같은 결과를 얻을 것입니다:

```
TimeCurrent = 2009.06.11 14:16:23
ArraySize(Time) = 10
TimeAsSeries[0] = 2009.06.11 14:00:00
TimeAsSeries[1] = 2009.06.11 13:00:00
TimeAsSeries[2] = 2009.06.11 12:00:00
TimeAsSeries[3] = 2009.06.11 11:00:00
TimeAsSeries[4] = 2009.06.11 10:00:00
TimeAsSeries[5] = 2009.06.11 09:00:00
TimeAsSeries[6] = 2009.06.11 08:00:00
TimeAsSeries[7] = 2009.06.11 07:00:00
TimeAsSeries[8] = 2009.06.11 06:00:00
TimeAsSeries[9] = 2009.06.11 05:00:00

ArrayNotSeries[9] = 2009.06.11 14:00:00
ArrayNotSeries[8] = 2009.06.11 13:00:00
ArrayNotSeries[7] = 2009.06.11 12:00:00
ArrayNotSeries[6] = 2009.06.11 11:00:00
ArrayNotSeries[5] = 2009.06.11 10:00:00
ArrayNotSeries[4] = 2009.06.11 09:00:00
ArrayNotSeries[3] = 2009.06.11 08:00:00
ArrayNotSeries[2] = 2009.06.11 07:00:00
ArrayNotSeries[1] = 2009.06.11 06:00:00
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

출력에서 볼 수 있듯이 TimeAsSeries 배열의 인덱스가 증가함에 따라 인덱스의 시간 값이 감소하는데, 즉, 현재에서 과거로 이동합니다. ArrayNotSeries의 경우 결과가 다른데, 인덱스가 증가함에 따라 과거에서 현재로 이동합니다.

See Also

ArrayIsDynamic, ArrayGetAsSeries, ArraySetAsSeries, ArrayIsSeries

© 2000-2025, MetaQuotes Ltd.

배열, 버퍼 및 시계열 인덱싱 방향

모든 배열 및 지표 버퍼의 기본 인덱스는 왼쪽에서 오른쪽으로입니다. 첫 번째 요소의 인덱스는 항상 0입니다. 따라서 색인 0을 가진 배열 또는 지표 버퍼의 첫 번째 요소는 기본적으로 맨 왼쪽 위치에 있고 마지막 요소는 맨 오른쪽 위치에 있습니다.

지표 버퍼는 클라이언트 터미널에서 크기를 관리하여 지표가 계산되는 막대의 수에 항상 일치하도록 하는 유형의 동적 배열입니다. SetIndexBuffer() 함수를 사용하여 일반적인 유형의 동적 이중 배열이 지표 버퍼로 할당됩니다.. 지표 버퍼는 ArrayResize() 함수를 사용하여 크기를 설정할 필요가 없습니다 – 이 작업은 터미널의 실행 시스템에서 수행합니다.

시계열은 역색인이 있는 배열입니다. 즉, 시계열의 첫 번째 요소는 오른쪽 끝에 있고 마지막 요소는 왼쪽 끝에 있습니다. 가격 내역 데이터를 저장하고 시간 정보를 포함하는 시계열은 시계열의 오른쪽 끝에 배치되며 가장 오래된 데이터는 왼쪽 끝에 배치됩니다.

따라서 인덱스가 0인 시계열 요소에는 심볼의 최신 따옴표에 대한 정보가 포함됩니다. 시계열에 일일 시간대의 데이터가 포함된 경우 현재 완료되지 않은 날의 데이터는 0 위치에 위치하며 인덱스 1의 위치에는 어제 데이터가 포함됩니다.

인덱싱 방향 변경

함수 ArraySetAsSeries()를 사용하면 동적 배열의 요소에 액세스하는 방법을 변경할 수 있습니다; 시스템 메모리에 저장된 데이터의 실제 순서는 변경되지 않습니다. 이 함수는 단순히 배열 요소의 주소 지정 방법을 변경하므로 함수 ArrayCopy()를 사용하여 한 배열을 다른 배열로 복사할 때 소스 배열의 인덱싱 방향에 따라 수신자 배열의 내용이 달라지지 않습니다.

정적 분포 배열의 인덱싱 방향은 변경할 수 없습니다. 배열이 함수에 매개 변수로 전달된 경우에도 이 기능 내에서 인덱싱 방향을 변경하려고 하면 아무런 효과가 없습니다.

지표 버퍼의 경우, 일반적인 배열과 마찬가지로 인덱싱 방향도 역방향으로 설정할 수 있습니다. 즉, 지표 버퍼의 0 포지션을 참조하는 것은 해당 지표 버퍼의 마지막 값을 참조하는 것이며 이는 최신 막대의 지표 값에 해당합니다. 그래도 표시 막대의 실제 위치는 변경되지 않습니다.

지표의 가격 데이터 수신

각 맞춤형 지표에는 지표 버퍼의 값을 계산하는 데 필요한 가격 데이터가 전달되는 OnCalculate() 함수가 반드시 포함되어야 합니다. ArrayGetAsSeries() 함수를 사용하여 이러한 통과된 배열의 인덱싱 방향을 확인할 수 있습니다.

배열 함수로 전달해 가격 데이터를 반영합니다, 즉, 이러한 배열에는 시계열의 심볼이 있으며 이러한 배열을 확인할 때 함수 ArrayIsSeries()가 참을 반환합니다. 단, 어떤 경우에도 지수화 명령은 ArrayGetAsSeries()함수에 의해서만 확인해야 합니다.

기본값에 종속되지 않으려면 사용할 어레이에 대해 무조건 ArraySetAsSeries()를 호출하고 필요한 방향을 설정해야 합니다.

가격 데이터 및 지표 값 수신

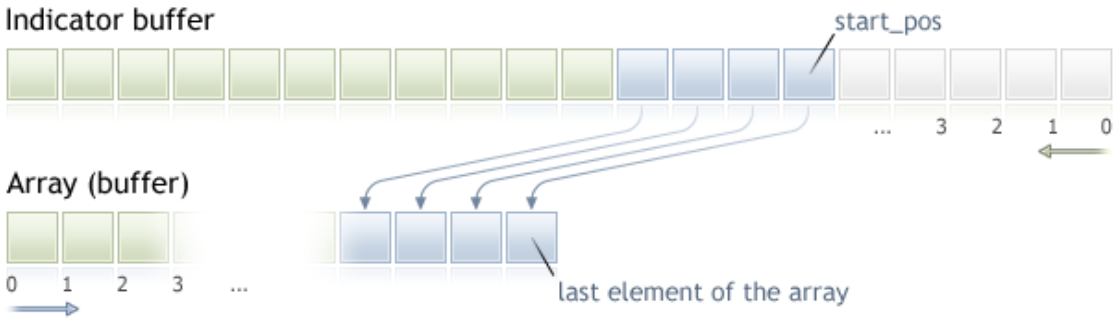
Expert Advisors, 지표 및 스크립트에서 모든 배열의 기본 인덱싱 방향은 왼쪽에서 오른쪽으로입니다. 필요한 경우 모든 mql5 프로그램에서 심볼 및 타임프레임에 대한 시계열 값뿐 아니라 심볼 및 타임프레임에 대해 계산된 값도 요청할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

Copy...() 함수를 사용하여 다음을 수행합니다:

- CopyBuffer – 지표 버퍼 값을 이중 유형의 배열로 복사;
- CopyRates – 일련의 구조에 가격 이력을 복사 MqlRates;
- CopyTime – 시간 값을 datetime 유형의 배열로 복사;
- CopyOpen – 오픈 값을 double 유형의 배열로 복사;
- CopyHigh – 고가를 double 유형의 배열로 복사;
- CopyLow – 저가를 double 유형의 배열로 복사;
- CopyClose – 종가를 double 유형의 배열로 복사;
- CopyTickVolume – 틱 거래량을 long 유형의 배열로 복사;
- CopyRealVolume – 자본 량을 long 유형의 배열로 복사;
- CopySpread – 스프레드 내역을 int 유형의 배열로 복사;

이 모든 함수들은 유사하게 작동합니다. CopyBuffer()의 예에서 데이터 수집 메커니즘을 살펴보겠습니다. 이는 요청된 데이터의 인덱싱 방향이 시계열 방향임을 암시하며, 인덱스가 0인 위치는 현재 완료되지 않은 막대의 데이터를 저장합니다. 이러한 데이터에 액세스하려면 필요한 양의 데이터를 수신자 배열 (예: 배열 *버퍼*)로 복사해야 합니다.



복사할 때 원본 배열의 시작 포지션을 지정해야 하며, 이 위치에서 수신자 배열로 데이터를 복사할 수 있습니다. 성공하면 지정된 개수의 요소가 소스 배열(이 경우 지표 버퍼)에서 수신자 배열로 복사됩니다. 수신자 배열에 설정된 인덱싱 값에 관계없이 복사는 항상 위 그림에 표시된 대로 수행됩니다.

가격 데이터가 반복 횟수가 많은 루프에서 처리될 것으로 예상되는 경우 IsStopped() 함수를 사용하여 강제 프로그램 종료 사실을 확인하는 것이 좋습니다:

```
int copied=CopyBuffer(ma_handle,// 지표 핸들
                      0,        // 지표 버퍼의 인덱스
                      0,        // 복사를 위한 시작 포지션
                      number,   // 복사할 값의 수
                      Buffer    // 값을 수령하는 배열
                      );
if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
  {
   //--- k 인덱스용 값 가져오기
   double value=Buffer[k];
```

© 2000-2025, MetaQuotes Ltd.

```
   // ...
   // 값과 함께 작동
   k++;
  }
```

예:

```
input int per=10; // 지수의 주기
int ma_handle;    // 지표 핸들
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// indicator handle
                         0,        // 지표 버퍼 인덱스
                         0,        // 복사할 시작 포지션
                         10,       // 복사를 위한 값 번호
                         ema       // 배열을 수령하는 값
                         );
   if(copied<0) return;
// .... 추가 코드
  }
```

더 보기

[데이터 액세스 구성]

© 2000-2025, MetaQuotes Ltd.

데이터 액세스 구성

이 섹션에서는 가격 데이터(시계열)의 획득, 저장 및 요청과 관련된 질문을 고려합니다.

트레이드 서버에서 데이터 수신

MetaTrader 5 터미널에서 가격 데이터를 이용할 수 있으려면 먼저 데이터를 받아 처리해야 합니다. 데이터를 수신하려면 MetaTrader 5 트레이드 서버에 연결해야 합니다. 데이터는 터미널의 요청에 따라 서버에서 분 막대의 팩킹된 블록 형태로 수신됩니다.

데이터에 대한 서버 참조 메커니즘은 요청이 어떻게 시작되었는지에 따라 달라지지 않습니다. 사용자가 차트나 MQL5 언어로 프로그램을 탐색할 때 그렇습니다.

중간 데이터 저장

서버에서 수신된 데이터는 자동으로 압축이 해제되고 HCC 중간 형식으로 저장됩니다. 각 심볼의 데이터는 다음과 같은 별도의 폴더에 기록됩니다: *terminal_directory*\bases*server_name*\history*symbol_name*. 예를 들어, MetaQuotes-Demo 서버에서 수신된 EURUSD의 데이터는 다음 위치에 저장됩니다. *terminal_directory*\bases\MetaQuotes-Demo\history\EURUSD\.

데이터는 확장자가 .hcc인 파일에 기록됩니다. 각 파일에는 1년 동안 분 막대의 데이터가 저장됩니다. 예를 들어, EURUSD 폴더에 있는 2009.hcc 파일은 2009년도의 EURUSD의 분 막대를 포함합니다. 이러한 파일은 모든 기간에 대한 가격 데이터를 준비하는 데 사용되며 직접 액세스하기 위한 것은 아닙니다.

중간 데이터 중 필요한 시간에 데이터 가져오기

중간 HCC 파일은 HC 형식의 요청된 기간에 대한 건물 가격 데이터의 데이터 소스로 사용됩니다. HC 형식의 데이터는 빠른 액세스를 위해 최대한으로 준비된 시계열입니다. 차트 또는 MQL5 프로그램의 요청에 따라 생성됩니다. 데이터 볼륨은 "차트의 최대 막대" 매개 변수의 값을 초과하면 안 됩니다. 데이터는 나중에 사용할 수 있도록 hc 확장자로 파일에 저장됩니다.

리소스를 절약하기 위해 필요한 경우에만 일정 기간의 데이터를 RAM에 저장하고 저장합니다. 오랫동안 호출하지 않으면 RAM에서 해제되어 파일에 저장됩니다. 각 기간에 대해 데이터는 다른 기간에 대한 준비 데이터가 있는지 여부에 관계없이 준비됩니다. 데이터 형성 및 액세스 규칙은 모든 기간 동안 동일합니다. 즉, HCC에 저장된 단위 데이터가 1분(M1)이라는 사실에도 불구하고, HCC 데이터의 가용성은 동일한 볼륨에서 HC와 같은 M1 기간에 데이터 가용성을 의미하는 것은 아닙니다.

서버로부터 새로운 데이터를 수신하면 모든 기간 동안 HC 형식의 중고 가격 데이터의 자동 업데이트가 요구됩니다. 또한 계산을 위한 입력 데이터로 암시적으로 사용하는 모든 지표가 재계산됩니다.

매개 변수 "차트의 최대 막대"

"차트의 최대 막대" 매개 변수는 차트, 지표 및 mql5 프로그램에서 사용할 수 있는 HC 형식의 막대 수를 제한합니다. 이는 사용 가능한 모든 기간 동안 유효하며 먼저 시스템 리소스를 절약하는 데 사용됩니다.

이 매개 변수의 큰 값을 설정할 때, 작은 시간대에 대한 깊은 기록 가격 데이터를 사용할 수 있는 경우 시계열 및 지표 버퍼를 저장하는 데 사용되는 메모리가 수백 메가바이트가 되어 클라이언트 터미널 프로그램의 RAM 제한에 도달할 수 있다는 점을 기억해야 합니다(MS Windows 32비트 애플리케이션의 경우 2Gb).

© 2000-2025, MetaQuotes Ltd.

클라이언트 터미널을 재시작한 후 "차트의 최대 막대" 변경 내용이 적용됩니다. 이 매개 변수를 변경하면 서버를 자동으로 참조하여 추가 데이터를 확인하거나 시계열의 추가 막대를 형성하지 않습니다. 서버에서 추가 가격 데이터를 요청하고, 데이터가 없는 영역으로 차트 스크롤하거나 mql5 프로그램에서 데이터를 요청하는 경우 새로운 제한 사항을 고려하여 시계열을 업데이트합니다.

서버에서 요청한 데이터 볼륨은 "차트의 최대 막대" 매개 변수를 고려하여 이 기간 동안 필요한 막대 수에 해당합니다. 이 매개 변수에 의해 설정된 제한은 엄격하지 않으며 경우에 따라 특정 기간 동안 사용 가능한 막대 수가 현재 매개 변수 값보다 약간 많을 수 있습니다.

데이터 가용성

HCC 형식 또는 HC 형식 사용을 위해 준비된 데이터가 있다고 해서 이러한 데이터가 차트에 표시되거나 MQL5 프로그램에 사용될 수 있는 절대 가용성을 나타내는 것은 아닙니다.

mql5 프로그램에서 가격 데이터 또는 지표 값에 액세스하는 경우 특정 시점 또는 특정 시점부터의 가용성이 보장되지 않는다는 점을 기억해야 합니다. 리소스를 절약하기 위해 mql5 프로그램에 필요한 전체 데이터 복사본이 MetaTrader 5에 저장되지 않고 터미널 데이터베이스에 대한 직접 액세스만 제공된다는 사실과 연결됩니다.

모든 기간에 대한 가격 기록은 HCC 형식의 공통 데이터로 구축되며, 서버의 모든 데이터 업데이트는 모든 기간에 대한 데이터 업데이트와 지표 재계산으로 이어집니다. 이러한 데이터 액세스는 잠시 전이라도 종료할 수 있습니다.

터미널 데이터와 서버 데이터의 동기화

mql5 프로그램은 어떤 심볼과 시간대의 데이터도 호출할 수 있기 때문에 필요한 시계열 데이터가 터미널에 아직 형성되지 않았거나 필요한 가격 데이터가 트레이드 서버와 동기화되지 않았을 가능성이 있습니다. 이 경우에는 지연 시간을 예측하기가 어렵습니다.

"아무것도 하지 않는" 루프를 사용하는 알고리즘은 최상의 솔루션이 아닙니다. 이 경우 스크립트만 예외입니다. 스크립트에는 이벤트 처리 기능이 없기 때문에 다른 알고리즘을 선택할 수 없기 때문입니다. 커스텀 지표의 경우, 그러한 알고리즘은 모든 지표의 계산 종료 및 심볼의 가격 데이터의 다른 취급으로 이어지기 때문에 다른 "아무 것도 하지 않는" 루프는 강력히 권장하지 않습니다.

Expert Advisor 및 지표의 경우 이벤트 모델을 사용하여 처리하는 것이 좋습니다. OnTick() 또는 OnCalculate() 이벤트를 처리하는 동안 필요한 시계열에 대한 데이터 수신이 실패한 경우 다음 핸들러 호출 시 액세스 가용성에 의존하여 이벤트 핸들러를 종료해야 합니다.

기록 추가를 위한 스크립트 예제

트랜잭션 서버에서 선택한 심볼에 대한 기록을 수신하기 위한 요청을 실행하는 스크립트의 예를 살펴보겠습니다. 이 스크립트는 선택한 심볼의 차트에서 실행하기 위한 것이며, 위에서 언급한 바와 같이 가격 데이터는 포장된 1분 데이터로 트레이드 서버에서 수신되고, 여기에서 사전 정의된 시계열은 구성되기 때문에 기간은 문제가 되지 않습니다.

데이터 수신과 관련된 모든 작업에 별도의 CheckLoadHistory 함수(symbol, timeframe, start_date):

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckLoadHistory() 함수는 모든 프로그램(Expert Advisor, 스크립트 또는 지표)에서 호출할 수 있는 범용 기능으로 설계되었으므로 필요한 가격 기록의 시작을 나타내려면 심볼 이름, 기간 및 시작 날짜의 세 가지 입력 매개 변수가 필요합니다.

누락된 기록을 요청하기 전에 함수 코드에 필요한 점검을 삽입하십시오. 먼저 심볼 이름과 주기 값이 올바른지 확인해야 합니다:

```
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
```

그런 다음 해당 심볼이 MarketWatch 창에서 사용 가능한지 확인합니다. 즉, 요청을 트레이드 서버로 전송할 때 심볼의 기록을 사용할 수 있습니다. If there is no such a symbol in MarketWatch, add it using the SymbolSelect() function.

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {f
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
```

이제 표시된 심볼/주기에 대해 사용 가능한 기록의 시작 날짜를 수신해야 합니다. CheckLoadHistory()로 전달된 입력 매개 변수 시작 날짜의 값이 사용 가능한 기록 내에 있을 수 있습니다. 그러면 트레이드 서버에 대한 요청이 필요하지 않습니다. 순간 현재 기호 주기의 첫 번째 날짜를 얻기 위해 SeriesInfoInteger() 함수를 SERIES_FIRSTDATE 변경자와 함께 사용합니다.

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

다음으로 중요한 검사는 함수가 호출되는 프로그램의 유형을 확인하는 것입니다. 지표에서 동일한 기간의 시계열 업데이트 요청을 보내는 것은 바람직하지 않습니다. 지표와 동일한 심볼 주기에 대한 데이터 요청이 바람직하지 않은 것은 지표가 작동하는 동일한 스레드에서 기록 데이터 업데이트가 수행된다는 사실에 의해 조절됩니다. 따라서 교착 상태가 발생할 가능성이 높습니다. 이를 확인하려면 MQL5InfoInteger() 함수를 MQL5_PROGRAM_TYPE 변경자와 함께 사용하십시오.

```
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
      return(-4);
```

모든 검사가 성공적으로 통과되었다면 거래 서버를 참조하지 않고 마지막 검사를 시도합니다. 먼저 HCC 형식의 분 데이터를 사용할 수 있는 시작 날짜를 알아보겠습니다. SERIES_TERMINAL_FIRSTDATE 변경자와 함께 SeriesInfoInteger() 함수를 사용하여 이 값을 요청하고 start_date 매개 변수의 값과 다시 비교합니다.

```
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- 시계열을 빌드하기 위해 로드된 데이터가 있습니다.
      if(first_date>0)
        {
         //--- 시계열 강제 생성
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- 날짜 점검
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
   }
```

모든 검사 후에도 실행 스레드가 CheckLoadHistory() 함수의 본문에 남아 있다면 거래 서버에 누락된 가격 데이터를 요청할 필요가 있음을 의미합니다. 먼저 TerminalInfoInteger() 함수를 사용하여 "차트의 최대 막대" 값을 반환합니다.:

```
  int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

추가 데이터 요청을 방지하기 위해 필요합니다. 그런 다음 SERIES_SERVER_FIRSTDATE 변경자와 함께 이미 알려진 SeriesInfoInteger() 함수를 사용하여 거래 서버의 심볼 내역에서 첫 번째 날짜를 찾습니다.

```
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
```

변수가 값을 수신하거나 사용자가 루프 실행을 종료할 때까지 5밀리초의 짧은 지연으로 함수가 루프에서 호출됩니다(IsStopped()는 *true를 반환합니다*, 이러한 경우). 거래 서버에 가격 데이터를 요청하는 시작 날짜의 정확한 값을 표시하겠습니다.

```
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("주의: 첫번째 서버 날짜 ",first_server_date," for ",
symbol," 첫 번째 시리즈 날짜와 일치하지 않음 ",first_date);
```

서버의 시작 날짜 *first_server_date*가 HCC 형식의 심볼 시작 날짜 *first_date* 보다 낮으면 해당 항목이 저널에 출력됩니다.

이제 우리는 거래 서버에 누락된 가격 데이터를 요청할 준비가 되었습니다. 루프 형태로 요청하고 차체 작성 시작:

```
   while(!IsStopped())
     {
      //1. HHC로 재구성된 시계열과 중간 기록 사이의 동기화를 기다립니다
      //2. 이 시계열의 현재 개수의 막대를 수령
      //    막대가 Max_bars_in_chart보다 크면, 종료할 수 있고, 작업이 끝난 것입니다
      //3. re-built 시계열에서 시작 날짜를 first_date로 가져와서 start_date와 비교
      //    first_date가 start_date보다 낮으면 종료할 수 있으며, 작업이 끝난 것입니다
      //4. 서버로부터 내역의 새로운 부분 요청 - 사용 가능한 마지막 막대 번호 'bars'부터 시작하는
     }
```

처음 세 가지는 이미 알려진 방법으로 구현됩니다.

```
   while(!IsStopped())
     {
      //--- 1.시계열 재구성 프로세스가 끝날 때까지 대기
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- 2.막대가 몇 개나 있는지 요청
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         //--- 차트에 그릴 수 있는 막대 두 개 이상, 종료
         if(bars>=max_bars) return(-2);
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- 3. 시계열의 현재 시작 날짜를 반환
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // 시작 날짜가 요청된 날짜보다 이전이며 작업이 완료됨
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //4. 서버로부터 내역의 새 파트 요청 - 사용 가능한 마지막 막대 번호 'bars'부터 시작하는 1
     }
```

마지막 네 번째 지점은 왼쪽 -요청 내역. 서버를 직접 참조할 수는 없지만, HCC 형식의 기록이 충분하지 않을 경우 Copy-function가 서버에 요청을 자동으로 전송합니다. *first_date* 변수에서 첫 번째 시작 날짜의 시간은 요청 실행 정도를 평가하는 간단하고 자연스러운 기준이기 때문에, 가장 쉬운 방법은 CopyTime() 함수를 사용하는 것입니다.

시계열에서 데이터를 복사하는 기능을 호출할 때는 *start* 파라미터(가격 데이터를 복사해야 하는 막대의 번호)가 항상 사용 가능한 터미널 기록 내에 있어야 합니다. 막대가 100개뿐이라면 인덱스 500을 사용하여 막대 300개를 복사하는 것은 의미가 없습니다. 이러한 요청은 오류로 인식되며 처리되지 않습니다. 즉, 트레이드 서버에서 추가 기록이 로드되지 않습니다.

그것이 바로 우리가 *막대* 인덱스를 가진 막대부터 100개 그룹으로 막대를 복하는 이유입니다. 이를 통해 트레이드 서버에서 누락된 이력을 원활하게 로드할 수 있습니다. 실제로 서버가 크기 초과 기록을 전송하는 동안 요청된 100개보다 조금 더 많은 막대가 로드됩니다.

```
   int copied=CopyTime(symbol,period,bars,100,times);
```

복사 작업이 끝나면 복사된 요소의 수를 분석해야 합니다. 시도가 실패하면 *복제된* 값이 null이 되고 fail_cnt 카운터의 값이 1 증가합니다. 100회 시도 실패 후 기능 작동이 중지됩니다.

```
int fail_cnt=0;
...
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- 데이터 점검
      if(times[0]<=start_date)  return(0);  // 복사된 값이 더 작고 준비됨
      if(bars+copied>=max_bars) return(-2); // 막대가 차트에서 그릴 수 있는 것보다 많습니다
      fail_cnt=0;
     }
   else
     {
      //--- 연속해서 실패한 시도가 100회 이하
      fail_cnt++;
      if(fail_cnt>=100) return(-5);
      Sleep(10);
     }
```

따라서 각 실행 순간의 현재 상황에 대한 올바른 처리가 기능에서 구현될 뿐만 아니라 추가 정보를 얻기 위해 CheckLoadHistory() 함수를 호출한 후 처리할 수 있는 종료 코드도 반환됩니다. 예를 들어 다음과 같습니다:

```
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      case -1 : Print("알수없는 심볼 ",InpLoadedSymbol);                    break;
      case -2 : Print("차트에서 그릴 수 있는 것보다 더 많은 요청 막대"); break;
      case -3 : Print("사용자에 의해 실행이 중단됨");                      break;
      case -4 : Print("지표가 자체 데이터를 로드해서는 안됨.");           break;
      case -5 : Print("로딩 실패");                                      break;
      case  0 : Print("모든 데이터 로드됨");                                 break;
      case  1 : Print("시계열로 이미 사용 가능한 데이터면 충분함");    break;
      case  2 : Print("시계열은 사용 가능한 터미널 데이터로 구축됨");   break;
      default : Print("실행 결과 정의되지 않음");
     }
```

함수의 전체 코드는 요청 결과를 처리하는 모든 데이터에 대한 올바른 액세스 구성을 보여 주는 스크립트의 예에서 확인할 수 있습니다.

Code:

```
//+------------------------------------------------------------------+
//|                                                TestLoadHistory.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- 입력 매개 변수
input string          InpLoadedSymbol="NZDUSD";   // Symbol to be load
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;  // Period to be loaded
input datetime        InpStartDate=D'2006.01.01'; // Start date
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   Print("Start load",InpLoadedSymbol+","+GetPeriodName(InpLoadedPeriod),"from",InpSta
//---
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("알수없는 심볼 ",InpLoadedSymbol);             break;
      case -2 : Print("요청된 막대가 차트의 최대 막대를 초과함"); break;
      case -3 : Print("프로그램 정지됨");                          break;
      case -4 : Print("지표가 자체 데이터를 로드해서는 안됨");      break;
      case -5 : Print("로딩 실패");                                 break;
      case  0 : Print("로딩 성공");                                   break;
      case  1 : Print("이전에 로드됨");                            break;
      case  2 : Print("이전에 로드되고 빌트됨");                  break;
      default : Print("알수없는 결과");
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   datetime first_date;
   SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
   int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
   Print("First date ",first_date," - ",bars," bars");
//---
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
   datetime first_date=0;
   datetime times[100];
//--- 심볼 및 주기 점검
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
//--- Market Watch에서 심볼이 선택되었는지 확인
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- 데이터가 있는지 확인
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
//--- 지표인 경우 자체 데이터 로드를 요청하지 않음
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
      return(-4);
//--- 두번째 시도
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- 시계열을 빌드하기 위해 로드된 데이터가 있습니다.
      if(first_date>0)
        {
         //--- 시계열 강제 생성
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- 날짜 점검
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
//--- 터미널 옵션에서 차트의 최대 막대
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- 심볼 내역 정보 로드
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
//--- 로드 시작 날짜 지정
```

© 2000-2025, MetaQuotes Ltd.

```
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("주의: 첫번째 서버 날짜",first_server_date," for ",symbol,
            " 첫 번째 시리즈 날짜와 일치하지 않음 ",first_date);
//--- 데이터를 차근차근 로드
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- 시계열 제작 대기
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- 막대 제작 요청
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         if(bars>=max_bars) return(-2);
         //--- 첫번째 날짜 요청
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //--- 다음 파트의 복제 데이터 강제 로드
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- 데이터 점검
         if(times[0]<=start_date)  return(0);
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- 100회 이하의 시도 실패
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
         Sleep(10);
        }
     }
//--- 정지됨
   return(-3);
  }
//+------------------------------------------------------------------+
//| 주기의 문자열 값을 반환                                          |
//+------------------------------------------------------------------+
string GetPeriodName(ENUM_TIMEFRAMES period)
  {
   if(period==PERIOD_CURRENT) period=Period();
//---
   switch(period)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      case PERIOD_M1:  return("M1");
      case PERIOD_M2:  return("M2");
      case PERIOD_M3:  return("M3");
      case PERIOD_M4:  return("M4");
      case PERIOD_M5:  return("M5");
      case PERIOD_M6:  return("M6");
      case PERIOD_M10: return("M10");
      case PERIOD_M12: return("M12");
      case PERIOD_M15: return("M15");
      case PERIOD_M20: return("M20");
      case PERIOD_M30: return("M30");
      case PERIOD_H1:  return("H1");
      case PERIOD_H2:  return("H2");
      case PERIOD_H3:  return("H3");
      case PERIOD_H4:  return("H4");
      case PERIOD_H6:  return("H6");
      case PERIOD_H8:  return("H8");
      case PERIOD_H12: return("H12");
      case PERIOD_D1:  return("Daily");
      case PERIOD_W1:  return("Weekly");
      case PERIOD_MN1: return("Monthly");
     }
//---
   return("알수없는 주기");
  }
```

© 2000-2025, MetaQuotes Ltd.

SeriesInfoInteger

내역 데이터의 상태에 대한 정보를 반환합니다. 함수 호출에는 두 가지 종류가 있습니다.

속성 값을 직접 반환합니다.

```
long  SeriesInfoInteger(
   string                      symbol_name,     // 심볼명
   ENUM_TIMEFRAMES             timeframe,       // 주기
   ENUM_SERIES_INFO_INTEGER    prop_id,         // 속성 식별자
   );
```

함수 실행의 성공 여부에 따라 true 또는 false를 반환합니다.

```
bool  SeriesInfoInteger(
   string                      symbol_name,     // 심볼명
   ENUM_TIMEFRAMES             timeframe,       // 주기
   ENUM_SERIES_INFO_INTEGER    prop_id,         // 속성 ID
   long&                       long_var         // 정보를 가져오는 변수
   );
```

Parameters

symbol_name
  [in] 심볼명.

timeframe
  [in] 주기.

prop_id
  [in] 요청된 속성의 식별자, ENUM_SERIES_INFO_INTEGER 열거값.

long_var
  [out] 요청된 속성의 값을 저장할 변수.

반환 값

첫 번째 경우 long 타입의 값을 반환합니다.

두 번째 경우에는 지정된 속성을 사용할 수 있고 해당 값이 X 변수에 배치된 경우 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 더 많은 정보에 대해선, GetLastError()을 호출하십시오.

예:

```
void OnStart()
  {
//---
   Print("현재 심볼 기간의 총 막대 수 = ",
         SeriesInfoInteger(Symbol(),Period(),SERIES_BARS_COUNT));

   Print("현재 심볼 기간의 첫 번째 날짜 = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_FIRSTDATE));
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("서버의 심볼 기간 기록 중 첫 번째 날짜 = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_SERVER_FIRSTDATE));

   Print("심볼 데이터가 동기화됨 = ",
         (bool)SeriesInfoInteger(Symbol(),Period(),SERIES_SYNCHRONIZED));
  }
```

© 2000-2025, MetaQuotes Ltd.

Bars

지정된 심볼 및 기간에 대한 기록의 막대 수를 반환합니다. 함수 호출에는 두 가지 종류가 있습니다.

모든 내역 막대 요청

```
int  Bars(
   string           symbol_name,     // 심볼명
   ENUM_TIMEFRAMES  timeframe        // 기간
   );
```

선택한 시간 간격에 대한 내역 막대 요청

```
int  Bars(
   string           symbol_name,     // 심볼명
   ENUM_TIMEFRAMES  timeframe,       // 주기
   datetime         start_time,      // 시작 일시
   datetime         stop_time        // 종료 일시
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_time

[in] 첫 번째 요소에 해당하는 막대 시간.

stop_time

[in] 마지막 요소에 해당하는 막대 시간.

반환 값

start_time 및 stop_time 매개 변수가 정의된 경우 함수는 지정된 시간 간격의 막대 수를 반환하고, 그렇지 않은 경우 전체 막대 수를 반환합니다.

참고

지정된 파라미터가 있는 시계열 데이터가 Bars() 함수 호출 시간까지 터미널에 형성되지 않거나 함수 호출 순간까지 시계열 데이터가 거래 서버와 동기화되지 않으면 함수는 0 값을 반환합니다.

지정된 시간 간격의 막대 수를 요청할 때는 해당 간격 내에 속하는 열린 시간이 있는 막대만 고려됩니다. 예를 들어, 현재 요일이 토요일이고 start_time=last_tuesday 및 stop_time=last_friday인 W1 바의 수에 대해 요청이 이루어진 경우, W1 기간의 열린 시간이 항상 일요일이고 지정된 간격 내에 있지 않기 때문에 함수는 0을 반환합니다.

모든 기록 막대의 개수에 대한 샘플 요청:

```
   int bars=Bars(_Symbol,_Period);
   if(bars>0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("현재 심볼 주기에 대한 터미널 기록의 막대 수 = ",bars);
     }
   else  //사용가능한 막대 없음
     {
      //--- 심볼의 데이터가 서버의 데이터와 동기화되지 않을 수 있음
      bool synchronized=false;
      //--- 루프 카운터
      int attempts=0;
      // 동기화 대기 시간 5회 시도
      while(attempts<5)
        {
         if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
           {
            //--- 동기화 완료, 종료
            synchronized=true;
            break;
           }
         //--- 카운터 증가
         attempts++;
         //--- 다음 반복까지 10밀리초 대기
         Sleep(10);
        }
      //--- 동기화 후 루프 종료
      if(synchronized)
        {
         Print("현재 심볼 주기에 대한 터미널 기록의 막대 수 = ",bars);
         Print("현재 심볼 기간의 터미널 기록에서 첫 번째 날짜 = ",
               (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
         Print("서버의 심볼에 대한 기록의 첫 번째 날짜 = ",
               (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
        }
      //--- 데이터 동기화가 일어나지 않음
      else
        {
         Print("다음을 위한 막대 수를 가져오는 데 실패",_Symbol);
        }
     }
```

지정된 간격의 막대 수에 대한 샘플 요청:

```
int n;
datetime date1 = D'2016.09.02 23:55'; // 금요일
datetime date2 = D'2016.09.05 00:00'; // 월요일
datetime date3 = D'2016.09.08 00:00'; // 목요일
//---
n=Bars(_Symbol,PERIOD_H1,D'2016.09.02 02:05',D'2016.09.02 10:55');
Print("막대 수: ",n); // 출력: "막대 수: 8", H2 막대 is는 연산에서 고려되지만 H11은 그렇지
n=Bars(_Symbol,PERIOD_D1,date1,date2);
Print("막대 수: ",n); // 출력: "막대 수: 1", 단일 D1 (월요일) 막대의 오픈 시간이 간격 내에
```

© 2000-2025, MetaQuotes Ltd.

```
   n=Bars(_Symbol,PERIOD_W1,date2,date3);
   Print("막대 수: ",n); // 출력: "막대 수: 0", 단일 W1 바 오픈 시간이 지정된 간격 내에 속하지
```

더 보기

이벤트 처리 함수

© 2000-2025, MetaQuotes Ltd.

BarsCalculated

지정된 지표에 대해 계산된 데이터 수를 반환합니다.

```
int  BarsCalculated(
   int       indicator_handle,     // 지표 핸들
   );
```

Parameters

indicator_handle

[in] 지표 핸들, 해당 지표 함수에 의해 반환됩니다.

반환 값

지표 버퍼에서 계산된 데이터 양을 반환하거나 오류 발생 시 -1 (아직 계산되지 않은 데이터)

참고

이 기능은 지표 생성 후 즉시 지표 데이터를 가져와야 할 때 유용합니다(지표 핸들을 사용할 수 있음).

예:

```
void OnStart()
  {
   double Ups[];
//--- 배열에 대한 시계열 순서 설정
   ArraySetAsSeries(Ups,true);
//--- 프랙탈 지표에 대한 핸들 생성
   int FractalsHandle=iFractals(NULL,0);
//--- 에러 코드를 재설정
   ResetLastError();
//--- 지표값 복사 시도
   int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Sleep(50);
      for(i=0;i<100;i++)
        {
         if(BarsCalculated(FractalsHandle)>0)
            break;
         Sleep(50);
        }
      copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
      if(copied<=0)
        {
         Print("상부 프랙탈 복사 실패. Error = ",GetLastError(),
         "i = ",i,"    copied = ",copied);
         return;
        }
       else
         Print("상부 프랙탈 복사됨",
```

© 2000-2025, MetaQuotes Ltd.

```
         "i = ",i,"    copied = ",copied);
     }
   else Print("상부 프랙탈 복사됨. ArraySize = ",ArraySize(Ups));
  }
```

© 2000-2025, MetaQuotes Ltd.

IndicatorCreate

이 함수는 MqlParam 유형의 매개 변수 베얄ㅇ,ㄹ 기반으로 생성된 지정된 기술 지표의 핸들을 반환합니다.

```
int  IndicatorCreate(
   string           symbol,                          // 심볼명
   ENUM_TIMEFRAMES  period,                          // 타임프레임
   ENUM_INDICATOR   indicator_type,                  // ENUM_INDICATOR 열거로부터의 지
   int              parameters_cnt=0,                // 매개 변수의 수
   const MqlParam&  parameters_array[]=NULL,         // 매개 변수 배열
   );
```

Parameters

symbol

[in] 지표가 계산되는 데이터의 심볼의 이름. NULL은 현재 심볼을 의미합니다.

period

[in] 타임프레임 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있으며, 0은 현재 타임프레임 범위를 의미합니다.

indicator_type

[in] 지표 유형, ENUM_INDICATOR 열거값 중 하나일 수 있음.

parameters_cnt

[in] parameters_array[] 배열에 전달된 매개 변수의 수. 배열 요소에는 특수 구조 유형 MqlParam이 있습니다. 기본적으로 0 매개 변수는 전달되지 않습니다. 매개 변수 수를 0이 아닌 값으로 지정할 경우 매개 변수 *parameters_array*는 필수입니다. 매개 변수는 64개까지만 전달할 수 있습니다.

parameters_array[]=NULL

[in] MqlParam type 유형의 배열로, 요소에는 기술적 지표의 각 입력 매개 변수 유형과 값이 포함됩니다.

반환 값

지정된 기술적 지표의 핸들을 반환하며, 장애가 발생한 경우 INVALID_HANDLE를 반환합니다.

참고

IND_CUSTOM 타입의 지표 핸들이 생성되면, *유형* 필드-입력 매개 변수 *parameters_array*의 배열 중 첫 번째 요소의-는 ENUM_DATATYPE 열거의 TYPE_STRING 값을 가져야 하며 첫번째 요소의 *string_value* 필드는 커스텀 지표의 이름을 포함해야 합니다. 커스텀 지표는 컴파일(EX5 확장자 파일)해야 하며 클라이언트 터미널의 MQL5/Indicators 디렉토리 또는 하위 디렉토리에 있어야 합니다.

테스트가 필요한 지표는 해당 파라미터가 constant string를 통해 설정된 경우 iCustom() 함수의 호출을 통해 자동으로 정의됩니다. 다른 모든 경우 (지표 이름을 설정하는 파라미터에서 IndicatorCreate() 함수 사용 또는 비상수 문자열 사용)에는 속성 #property tester_indicator가 필요합니다:

```
#property tester_indicator "indicator_name.ex5"
```

커스텀 지표에 첫 번째 형태의 호출이 사용되는 경우, 입력 파라미터를 전달할 때 어떤 데이터가 계산될지 마지막 파라미터로 추가로 지정할 수 있습니다. "적용 대상" 파라미터가 명시적으로 지정되지 않은 경우 기본 계산은 PRICE_CLOSE 값을 기반으로 합니다.

© 2000-2025, MetaQuotes Ltd.

例:

```
void OnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- create iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE);
   ArrayResize(params,4);
//--- set ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- set ma_shift
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- set ma_method
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- set applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- create MA
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- create iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA);
   ArrayResize(params,4);
//--- set fast ma_period
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- set slow ma_period
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- set smooth period for difference
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- set indicator handle as applied_price
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- create MACD based on moving average
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- use indicators
//--- . . .
//--- release indicators (first h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

© 2000-2025, MetaQuotes Ltd.

IndicatorParameters

지정된 핸들을 기준으로 지표의 입력 파라미터 수와 파라미터의 값 및 유형을 반환합니다.

```
int  IndicatorParameters(
   int               indicator_handle,     // 지표 핸들
   ENUM_INDICATOR&   indicator_type,       // 지표 유형 수신 변수
   MqlParam&         parameters[]          // 수신 매개 변수 배열
   );
```

Parameters

indicator_handle

[in] 계산된 파라미터의 수를 알아야 하는 지표의 핸들.

indicator_type

[out] 지표 유형을 기록할 ENUM_INDICATOR 유형인 경우의 변수.

parameters[]

[out] 지표 매개변수 목록이 기록될 MqlParam 유형의 값을 수신하기 위한 동적 배열 IndicatorParameters() 함수에 의해 배열 크기가 반환됩니다.

반환 값

지정된 핸들을 사용하는 지표의 입력 파라미터 수. 오류의 경우 -1을 반환합니다. 오류에 대한 더 많은 정보에 대해선 GetLastError() 함수를 호출하십시오.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- 차트의 창 수(최소 하나 이상의 기본 창이 항상 있음)
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- 차트 창을 살펴봅니다
   for(int w=0;w<windows;w++)
     {
      //--- 이 창/하위 창의 지표 수
      int total=ChartIndicatorsTotal(0,w);
      //--- 창의 모든 지표 표시
      for(int i=0;i<total;i++)
        {
         //--- 지표의 짧은 이름을 가져옵니다
         string name=ChartIndicatorName(0,w,i);
         //--- 지표 핸들 가져오기
         int handle=ChartIndicatorGet(0,w,name);
         //--- Add to log
         PrintFormat("Window=%d,  indicator #%d,  handle=%d",w,i,handle);
         //---
```

© 2000-2025, MetaQuotes Ltd.

```
         MqlParam parameters[];
         ENUM_INDICATOR indicator_type;
         int params=IndicatorParameters(handle,indicator_type,parameters);
         //--- 메시지의 헤더
         string par_info="Short name "+name+", type "
                         +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
         //---
         for(int p=0;p<params;p++)
           {
            par_info+=StringFormat("parameter %d: type=%s, long_value=%d, double_value
                                   p,
                                   EnumToString((ENUM_DATATYPE)parameters[p].type),
                                   parameters[p].integer_value,
                                   parameters[p].double_value,
                                   parameters[p].string_value
                                   );
           }
         Print(par_info);
        }
      //--- 창의 모든 지표에 대해 완료
     }
//---
  }
```

더 보기

ChartIndicatorGet()

© 2000-2025, MetaQuotes Ltd.

IndicatorRelease

이 함수는 지표 핸들을 제거하고 다른 사용자가 사용하지 않을 경우 지표의 계산 블록을 해제합니다.

```
bool  IndicatorRelease(
   int       indicator_handle     // 지표 핸들
   );
```

반환 값

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

참고

이 함수를 사용하면 더 이상 필요하지 않은 지표 핸들을 제거하여 메모리를 절약할 수 있습니다. 핸들이 즉시 제거되고 계산 블록이 잠시 후에 삭제됩니다(더 이상 호출되지 않는 경우).

전략 테스터에서 작업할 경우, the IndicatorRelease() 함수가 실행되지 않습니다.

예:

```
//+------------------------------------------------------------------+
//|                                      Test_IndicatorRelease.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 입력 매개 변수
input int                MA_Period=15;
input int                MA_shift=0;
input ENUM_MA_METHOD     MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- 지표 핸들을 저장
int MA_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 핸들 생성
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,PRICE_CLOSE);
//--- 글로벌 변수 삭제
   if(GlobalVariableCheck("MA_value"))
      GlobalVariableDel("MA_value");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
```

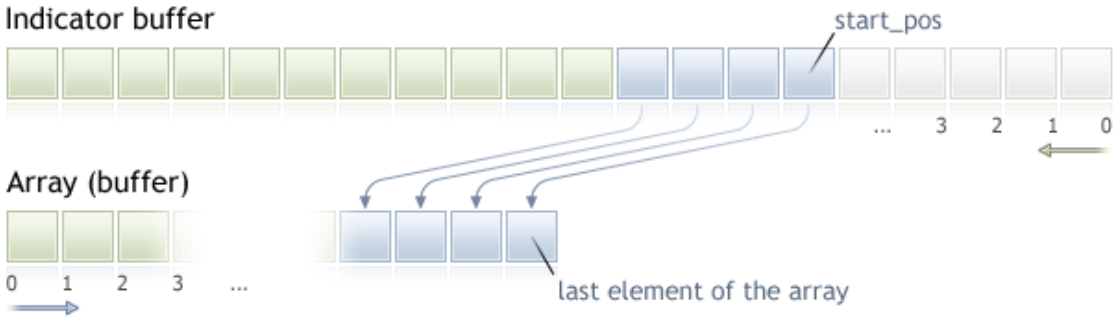
© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 글로벌 변수 값이 없는 경우
   if(!GlobalVariableCheck("MA_value"))
     {
      //--- 마지막 두 막대의 지표 값을 구함
      if(MA_handle!=INVALID_HANDLE)
        {
         //--- 지표 값에 대한 동적 배열
         double values[];
         if(CopyBuffer(MA_handle,0,0,2,values)==2 && values[0]!=EMPTY_VALUE)
           {
            //--- 마지막 막대에서 글로벌 변수 값으로 기억
            if(GlobalVariableSet("MA_value",values[0]))
              {
               //--- 지표의 핸들을 비움
               if(!IndicatorRelease(MA_handle))
                  Print("IndicatorRelease() 실패. Error ",GetLastError());
               else MA_handle=INVALID_HANDLE;
              }
            else
               Print("GlobalVariableSet 실패. Error ",GetLastError());
           }
        }
     }
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

CopyBuffer

특정 지표의 지정된 버퍼 데이터를 필요한 수량으로 가져옵니다.



시작 위치에서 복사된 데이터 요소(인덱스 buffer_num을 가진 지표 버퍼)의 카운트는 현재에서 과거로 수행됩니다. 즉, 시작 위치가 0이면 현재 막대(현재 막대에 대한 지표 값)를 의미합니다.

아직 알 수 없는 양의 데이터를 복사할 때는 CopyBuffer() 함수가 수신 배열의 크기를 복사된 데이터의 크기에 할당하려고 하므로 동적 배열을 buffer[]로 사용하는 것이 좋습니다. 지표 버퍼(SetIndexBufer() 함수가 지표 값을 저장하기 위해 미리 할당된 배열)를 buffer[] 수신인 배열로 사용하는 경우 부분 복사가 허용됩니다. 예를 들어 표준 터미널 패키지의 Awesome_Oscillator.mq5 커스텀 지표에서 확인할 수 있습니다.

지표 값의 일부를 다른 배열(비-지표 버퍼)에 복사해야 하는 경우 원하는 숫자가 복사되는 중간 어레이를 사용해야 합니다. 그런 다음, 필요한 수의 값을 이 중간 값에서 필요한 수신 배열의 위치로 요소별로 복사합니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyBuffer(
   int       indicator_handle,     // 지표 핸들
   int       buffer_num,           // 지표 버퍼 넘버
   int       start_pos,            // 시작 포지션
   int       count,                // 복사할 양
   double    buffer[]              // 복사할 대상 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyBuffer(
   int       indicator_handle,     // 지표 핸들
   int       buffer_num,           // 지표 버퍼 넘버
   datetime  start_time,           // 시작 일시
```

© 2000-2025, MetaQuotes Ltd.

```
   int       count,                // 복사할 양
   double    buffer[]              // 복사할 대상 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyBuffer(
   int       indicator_handle,     // 지표 핸들
   int       buffer_num,           // 지표 버퍼 넘버
   datetime  start_time,           // 시작 일시
   datetime  stop_time,            // 종료 일시
   double    buffer[]              // 복사할 대상 배열
   );
```

Parameters

indicator_handle

[in] 지표 핸들, 해당 지표 함수에 의해 반환됩니다.

buffer_num

[in] 지표 버퍼 번호.

start_pos

[in] 복사할 첫 번째 요소의 포지션.

count

[in] 복사할 데이터 수.

start_time

[in] 첫 번째 요소에 해당하는 막대 시간.

stop_time

[in] 마지막 요소에 해당하는 막대 시간.

buffer[]

[out] double 유형의 배열.

반환 값

오류 발생 시 복사된 데이터 개수 또는 -1을 반환합니다.

참고

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환합니다.

예:

```
//+------------------------------------------------------------------+
//|                                              TestCopyBuffer3.mq5 |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 입력 매개 변수
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- 지표 버퍼
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 연결
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Parameter AsSeries = ",AsSeries);
   Print("SetIndexBuffer()이후의 지표 버퍼는 시계열입니다 = ",
         ArrayGetAsSeries(MABuffer));
//--- 짧은 지표 이름을 설정
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- AsSeries로 설정 (입력 파라미터에 따라 다름)
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("Indicator buffer after ArraySetAsSeries(MABuffer,true); is a timeseries = ",
         ArrayGetAsSeries(MABuffer));
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
```

© 2000-2025, MetaQuotes Ltd.

```
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 모든 데이터가 계산되었는지 확인
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- 모든 데이터를 복사할 수 있는 것은 아닙니다
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- 마지막 값은 항상 복사됩니다
      to_copy++;
     }
//--- 복사 시도
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

위의 예에서는 지표 버퍼가 동일한 심볼/주기에 있는 지표의 다른 지표 버퍼 값으로 채워지는 방법을 보여 줍니다.

데이터를 요청하는 내역의 자세한 예는 객체 바인딩 방법 섹션에서 참조하십시오. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

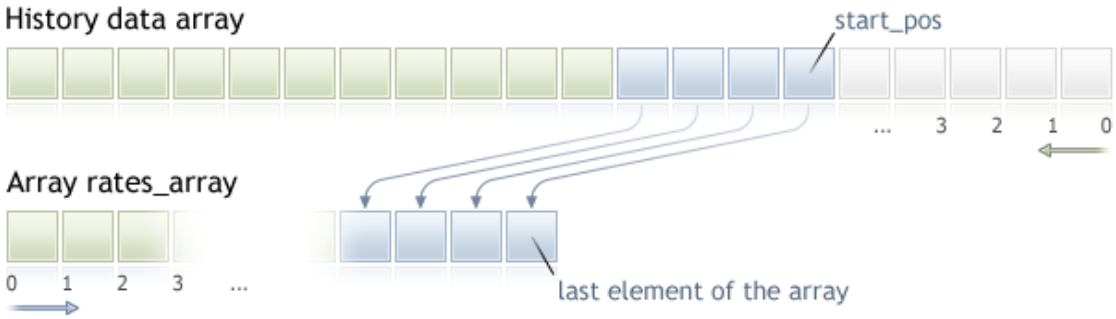
더 보기

Properties of Custom Indicators, SetIndexBuffer

© 2000-2025, MetaQuotes Ltd.

CopyRates

지정된 수량의 지정된 심볼 주기의 MqlRates 구조에 대한 기록 데이터를 rates_array 배열로 가져옵니다. 복사된 데이터의 요소 순서는 현재에서 과거로이며, 즉, 시작 위치가 0이면 현재 막대를 의미합니다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyRates(
   string           symbol_name,       // 심볼명
   ENUM_TIMEFRAMES  timeframe,         // 주기
   int              start_pos,         // 시작 포지션
   int              count,             // 복사할 데이터 카운트
   MqlRates         rates_array[]      // 복사할 대상 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyRates(
   string           symbol_name,       // 심볼명
   ENUM_TIMEFRAMES  timeframe,         // 주기
   datetime         start_time,        // 시작 일시
   int              count,             // 복사할 데이터 카운트
   MqlRates         rates_array[]      // 복사할 대상 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyRates(
   string           symbol_name,       // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,          // 주기
   datetime          start_time,         // 시작 일시
   datetime          stop_time,          // 종료 날짜 및 시간
   MqlRates          rates_array[]       // 복사할 대상 배열
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_time

[in] 복사할 첫 번째 요소의 막대 시간.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

stop_time

[in] 마지막 복사 요소에 해당하는 막대 시간.

rates_array[]

[out] MqlRates 유형의 배열.

반환 값

오류가 발생한 경우 복사된 요소의 수를 반환하거나 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

예:

```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Bars copied: "+copied);
      string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
      string out;
      int size=fmin(copied,10);
      for(int i=0;i<size;i++)
        {
         out=i+":"+TimeToString(rates[i].time);
         out=out+" "+StringFormat(format,
                                  rates[i].open,
                                  rates[i].high,
                                  rates[i].low,
                                  rates[i].close,
                                  rates[i].tick_volume);
         Print(out);
        }
     }
   else Print("심볼에 대한 기록 데이터를 가져오지 못했습니다. ",Symbol());
  }
```

객체 바인딩 방법 섹션에서 기록 데이터 요청의 자세한 예를 참조하십시오. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

더 보기

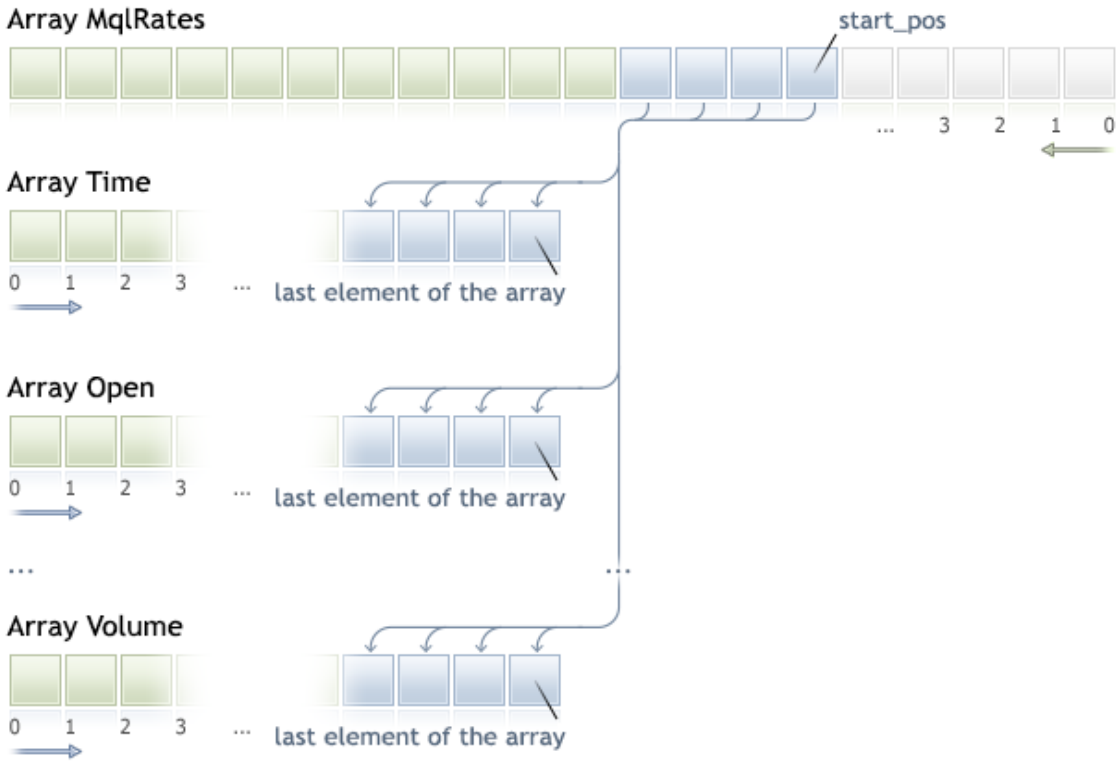
© 2000-2025, MetaQuotes Ltd.

Structures and Classes, TimeToString, StringFormat

© 2000-2025, MetaQuotes Ltd.

CopySeries

지정된 심볼-기간 및 지정된 양만큼 MqlRates 구조에서 동기화된 시계열을 가져옵니다. 데이터는 지정된 배열 세트로 수신됩니다. 요소는 현재부터 과거로 카운트다운 됩니다. 즉 시작 위치가 0이면 현재 바를 의미합니다.



복사할 데이터 양을 알 수 없는 경우 수신 배열로 동적 배열을 사용하는 것이 좋습니다. 왜냐하면 데이터의 양이 배열이 포함할 수 있는 양을 초과하면 요청된 모든 데이터에 맞게 배열을 재배포하려는 시도가 발생할 수 있기 때문입니다.

미리 정해진 양의 데이터를 복사해야 하는 경우 메모리가 불필요하게 재할당 되는 것을 막기 위해 통계적으로 할당된 버퍼를 사용하는 것이 좋습니다.

수신하는 배열의 속성(as_series=true 또는 as_series=false)은 무시됩니다. 복사하는 동안 가장 오래된 시계열 요소가 배열에 할당된 물리적 메모리의 시작 부분에 복사됩니다.

```
int  CopySeries(
   string           symbol_name,       // 심볼명
   ENUM_TIMEFRAMES  timeframe,         // 차트 주기
   int              start_pos,         // 시작 포지션
   int              count,             // 복사할 양
   ulong            rates_mask,        // 요청된 시리즈를 지정할 플래그 조합
   void&            array1[],          // 첫 번째 복사본 시계열의 데이터를 수신하는 배열
   void&            array2[]           // 두 번째 복사된 시계열의 데이터를 수신할 배열
   ...
   );
```

© 2000-2025, MetaQuotes Ltd.

패러미터

symbol_name

[in] 심볼.

timeframe

[in] 차트 주기

start_pos

[in] 첫번째 복사된 요소 인덱스

count

[in] 복사된 요소의 수

rates_mask

[in] ENUM_COPY_RATES 열거의 플래그 조합

array1, array2,...

MqlRates 구조에서 [out] 시계열을 수신하는 데 적합한 유형의 배열. 함수에 전달된 배열의 순서는 MqlRates 구조의 필드 순서와 일치해야 합니다.

Return Value

복사된 요소의 수, 오류가 발생한 경우는 -1

주의

요청된 데이터의 전체 간격이 서버에서 사용 가능한 데이터를 초과하는 경우에는 함수는 -1을 반환합니다. 요청된 데이터가 TERMINAL_MAXBARS(차트의 최대 바의 수)를 초과하는 경우에도 함수는 -1을 반환합니다.

지표에서 데이터를 요청할 때 요청된 시계열이 아직 구성되지 않았거나 서버에서 다운로드해야 하는 경우 함수는 즉시 -1을 반환합니다. 그러나 이렇게 하면 자체적으로 데이터 다운로드/구성이 시작됩니다.

터미널에 적절한 데이터가 로컬에 없으면 Expert Advisor 또는 스크립트에서 데이터를 요청할 때 서버에서 다운로드가 시작됩니다. 혹은 필요한 로컬 기록에서 데이터를 구성할 수 있지만 아직 준비되지 않은 경우 시계열이 시작됩니다. 함수는 제한 시간이 만료될 때까지 준비된 데이터를 반환하지만 기록 다운로드는 계속되고 함수는 이후의 유사한 요청에 대해 계속 데이터를 반환합니다.

CopySeries와 CopyRates의 차이

CopySeries 기능을 사용하면 한 번의 호출에서 필요한 시계열만을 지정된 배열로 가져올 수 있으며 모든 시계열 데이터는 동기화됩니다. 이는 특정 인덱스 N에 있는 결과 배열의 모든 값이 지정된 심볼/주기 쌍의 동일한 바에 속한다는 것을 의미합니다. 따라서 프로그래머는 바가 열리는 시간까지 수신된 모든 시계열의 동기화를 보장할 필요가 없습니다.

전체 시계열 집합을 MqlRates 배열로 반환하는 CopyRates와 달리 CopySeries 함수를 사용하면 프로그래머가 필요한 시계열만 별도의 배열로 가져올 수 있습니다. 이는 시계열 유형을 선택하는 플래그 조합을 지정하여 수행할 수 있습니다. 함수에 전달된 배열의 순서는 MqlRates 구조의 필드 순서와 일치해야 합니다.

```
struct MqlRates
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   datetime time;         // 주기 시작 시간
   double   open;         // 시가
   double   high;         // 주기내 고가
   double   low;          // 주가내 저가
   double   close;        // 종가
   long     tick_volume;  // 틱 볼륨
   int      spread;       // 스프레드
   long     real_volume;  // 교환 볼륨
  }
```

따라서 현재 Symbol/Timeframe의 마지막 100개 막대에 대한 *time*, *close* 및 *real_volume* 의 시계열의 값을 가져와야 하는 경우 다음의 호출을 사용해야 합니다:

```
datetime time[];
double   close[];
long     volume[];
CopySeries(NULL,0,0,100,COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_VOLUME_REAL,time,c
```

배열의 순서 주의 "*time*, *close*, *volume*" — 반드시 MqlRates 구조의 필드의 순서와 일치해야 합니다. *rates_mask*의 값의 순서는 중요하지 않습니다. mask는 다음과 같을 수 있습니다.

```
COPY_RATES_VOLUME_REAL|COPY_RATES_TIME|COPY_RATES_CLOSE
```

예:

```
//--- 입력 매개변수
input datetime InpDateFrom=D'2022.01.01 00:00:00';
input datetime InpDateTo  =D'2023.01.01 00:00:00';
input uint     InpCount   =20;
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
void OnStart(void)
  {
//--- MqlRates 가격 구조에서 시계열을 가져오는 배열 구조
   double   open[];
   double   close[];
   float    closef[];
   datetime time1[], time2[];
//--- 이중 배열에 종가 요청
   ResetLastError();
   int res1=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE, time1, close);
   PrintFormat("1. CopySeries  returns %d values. Error code=%d", res1, GetLastError()
   ArrayPrint(close);

//--- 이제 시가도 요청합니다; 종가에 float 배열 사용
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int res2=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_OPEN, time2, open,
   PrintFormat("2. CopySeries  returns %d values. Error code=%d", res2, GetLastError()
   ArrayPrint(closef);
//--- 수신산 데이터를 비교
   if((res1==res2) && (time1[0]==time2[0]))
     {
      Print("  | Time              |    Open      | Close double | Close float |");
      for(int i=0; i<10; i++)
        {
         PrintFormat("%d | %s |   %.5f    |   %.5f    |   %.5f   |",
                     i, TimeToString(time1[i]), open[i], close[i], closef[i]);
        }
     }
//--- 결과
      1. CopySeries가 20개 값을 반환. Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.(
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.(
      2. CopySeries가 20개 값을 반환. Error code=0
      [ 0] 1.06722 1.06733 1.06653 1.06520 1.06573 1.06649 1.06694 1.06675 1.06684 1.(
      [10] 1.06514 1.06557 1.06456 1.06481 1.06414 1.06394 1.06364 1.06386 1.06239 1.(
        | Time              |    Open      | Close double | Close float |
      0 | 2023.03.01 17:00 |   1.06660    |   1.06722    |   1.06722   |
      1 | 2023.03.01 18:00 |   1.06722    |   1.06733    |   1.06733   |
      2 | 2023.03.01 19:00 |   1.06734    |   1.06653    |   1.06653   |
      3 | 2023.03.01 20:00 |   1.06654    |   1.06520    |   1.06520   |
      4 | 2023.03.01 21:00 |   1.06520    |   1.06573    |   1.06573   |
      5 | 2023.03.01 22:00 |   1.06572    |   1.06649    |   1.06649   |
      6 | 2023.03.01 23:00 |   1.06649    |   1.06694    |   1.06694   |
      7 | 2023.03.02 00:00 |   1.06683    |   1.06675    |   1.06675   |
      8 | 2023.03.02 01:00 |   1.06675    |   1.06684    |   1.06684   |
      9 | 2023.03.02 02:00 |   1.06687    |   1.06604    |   1.06604   |
//---
  }
```

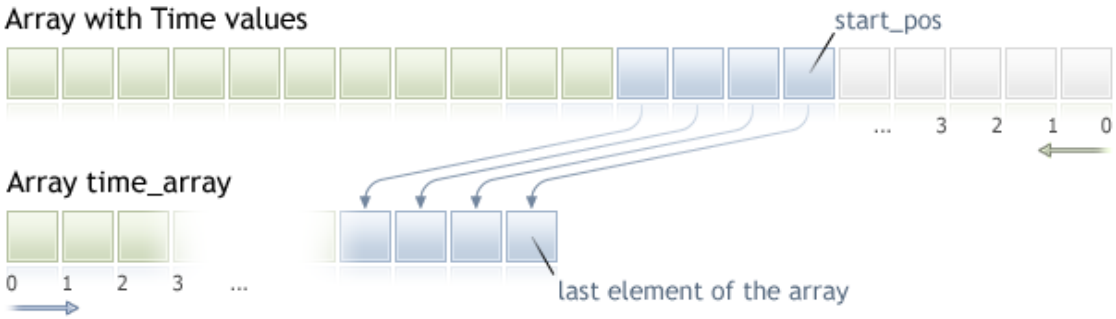
더 참조할 만한 내용

구조와 클래스, CopyRates

© 2000-2025, MetaQuotes Ltd.

CopyTime

이 함수는 지정된 양의 지정된 심볼-주기에 대한 막대 오프닝 시간의 time_array 기록 데이터를 가져옵니다. 다음 주문 요소에 유의해야 합니다: 현재에서 과거, 즉 0의 시작 위치는 현재 막대를 의미한다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyTime(
   string           symbol_name,     // 심볼명
   ENUM_TIMEFRAMES  timeframe,       // 주기
   int              start_pos,       // 시작 포지션
   int              count,           // 복제할 데이터 수
   datetime         time_array[]     // 오픈 시간을 복사할 대상 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyTime(
   string           symbol_name,     // 심볼명
   ENUM_TIMEFRAMES  timeframe,       // 주기
   datetime         start_time,      // 시작 일시
   int              count,           // 복제할 데이터 수
   datetime         time_array[]     // 오픈 시간을 복사할 대상 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyTime(
   string           symbol_name,     // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES  timeframe,       // 주기
   datetime         start_time,      // 시작 일시
   datetime         stop_time,       // 정지 일시
   datetime         time_array[]     // 오픈 시간을 복사할 대상 배열
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

start_time

[in] 복사할 첫 번째 요소의 시작 시간.

stop_time

[in] 복사할 마지막 요소에 해당하는 막대 시간.

time_array[]

[out] datetime 타입의 배열.

반환 값

오류 발생 시 복사된 데이터 개수 또는 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

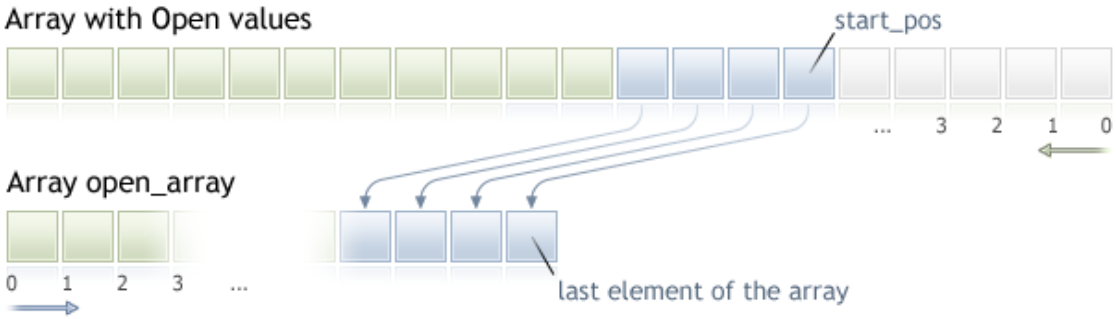
객체 바인딩 방법 섹션에서 기록 데이터 요청의 자세한 예를 참조하십시오. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyOpen

이 함수는 지정된 수량의 선택된 심볼-기간에 대한 바 오픈 가격의 이력 데이터를 open_array에 넣습니다. 요소 순서는 현재에서 과거까지입니다. 즉, 0의 시작 포지션은 현재 막대를 의미합니다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyOpen(
   string           symbol_name,     // 심볼명
   ENUM_TIMEFRAMES  timeframe,       // 주기
   int              start_pos,       // 시작 포지션
   int              count,           // 복제할 데이터 수
   double           open_array[]     // 시가를 복사하도록 배열을 지정
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyOpen(
   string           symbol_name,     // 심볼명
   ENUM_TIMEFRAMES  timeframe,       // 주기
   datetime         start_time,      // 시작 일시
   int              count,           // 복제할 데이터 수
   double           open_array[]     // 시가 막대용 타겟 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyOpen(
   string           symbol_name,     // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,       // 주기
   datetime          start_time,      // 시작 일시
   datetime          stop_time,       // 정지 일시
   double            open_array[]     // 시가 값들용 타겟 배열
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

start_time

[in] 복사할 첫 번째 요소의 시작 시간.

stop_time

[in] 복사할 마지막 요소의 시작 시간.

open_array[]

[out] double 유형의 배열.

반환 값

배열의 요소 수를 반환하거나 오류발생 시 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

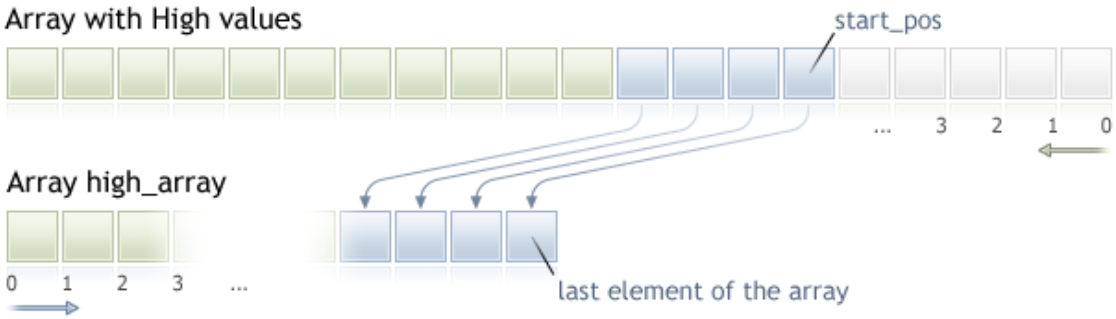
객체 바인딩 방법 섹션에서 기록 데이터 요청의 자세한 예를 참조하십시오. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyHigh

함수는 지정된 수량의 선택된 심볼-주기에 대해 가장 고점의 막대 가격의 기록 데이터를 high_array에 넣습니다. 요소 순서는 현재에서 과거까지입니다. 즉, 0의 시작 포지션은 현재 막대를 의미합니다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyHigh(
   string           symbol_name,      // 심볼명
   ENUM_TIMEFRAMES  timeframe,        // 주기
   int              start_pos,        // 포지션 시작
   int              count,            // 복제할 데이터 수
   double           high_array[]      // 복제할 타겟 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyHigh(
   string           symbol_name,      // 심볼명
   ENUM_TIMEFRAMES  timeframe,        // 주기
   datetime         start_time,       // 시작 일시
   int              count,            // 복제할 데이터 수
   double           high_array[]      // 복제할 타겟 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyHigh(
   string           symbol_name,      // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,        // 주기
   datetime          start_time,       // 시작 일시
   datetime          stop_time,        // 정지 일시
   double            high_array[]      // 복제할 타겟 배열
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

start_time

[in] 복사할 첫 번째 요소의 시작 시간.

stop_time

[in] 마지막 복사 요소에 해당하는 막대 시간.

high_array[]

[out] double 유형의 배열.

반환 값

오류 발생 시 복사된 데이터 개수 또는 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

예:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "High[i] 및 Low[i] 결과의 예"
#property description "랜덤 선택 막대용"

double High[],Low[];
//+------------------------------------------------------------------+
//| Get Low for specified bar index                                  |
//+------------------------------------------------------------------+
double iLow(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double low=0;
   ArraySetAsSeries(Low,true);
   int copied=CopyLow(symbol,timeframe,0,Bars(symbol,timeframe),Low);
   if(copied>0 && index<copied) low=Low[index];
   return(low);
  }
//+------------------------------------------------------------------+
//| 지정된 막대 지수용 고점 가져오기                                 |
//+------------------------------------------------------------------+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
  {
   double high=0;
   ArraySetAsSeries(High,true);
   int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
   if(copied>0 && index<copied) high=High[index];
   return(high);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 모든 틱에서 인덱스가 있는 막대의 고가 및 저가 값을 출력합니다
//--- 이 값은 틱이 도착한 두 번째 값과 같습니다
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
```

© 2000-2025, MetaQuotes Ltd.

```
            sec,iLow(Symbol(),0,sec));
  }
```

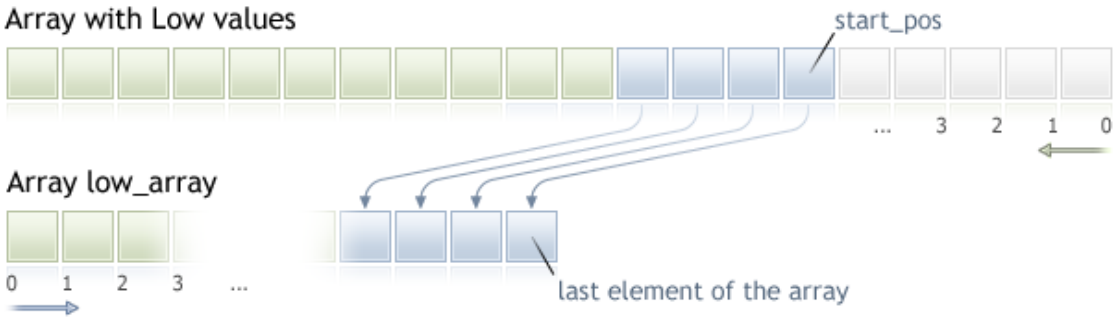
기록 데이터를 요청하는 자세한 예는 객체 바인딩 방법 섹션에서 참조하십시오.. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyLow

이 함수는 지정된 수량의 선택된 심볼-주기에 대한 가장 저점의 막대 가격의 기록 데이터를 low_array에 넣습니다. 요소 순서는 현재에서 과거까지입니다. 즉, 0의 시작 포지션은 현재 막대를 의미합니다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyLow(
   string           symbol_name,     // 심볼명
   ENUM_TIMEFRAMES  timeframe,       // 주기
   int              start_pos,       // 시작 포지션
   int              count,           // 복제할 데이터 수
   double           low_array[]      // 복제할 타겟 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyLow(
   string           symbol_name,     // 심볼명
   ENUM_TIMEFRAMES  timeframe,       // 주기
   datetime         start_time,      // 시작 일시
   int              count,           // 복제할 데이터 수
   double           low_array[]      // 복제할 타겟 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyLow(
   string           symbol_name,     // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,       // 주기
   datetime          start_time,      // 시작 일시
   datetime          stop_time,       // 정지 일시
   double            low_array[]      // 복제할 타겟 배열
   );
```

Parameters

symbol_name

[in] 심볼.

timeframe

[in] 주기.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

start_time

[in] 첫 번째 복사 요소에 해당하는 막대 시간.

stop_time

[in] 마지막 복사 요소에 해당하는 막대 시간.

low_array[]

[out] double 유형의 배열.

반환 값

오류 발생 시 복사된 데이터 개수 또는 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

객체 바인딩 방법 섹션에서 기록 데이터 요청의 자세한 예를 참조하십시오. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

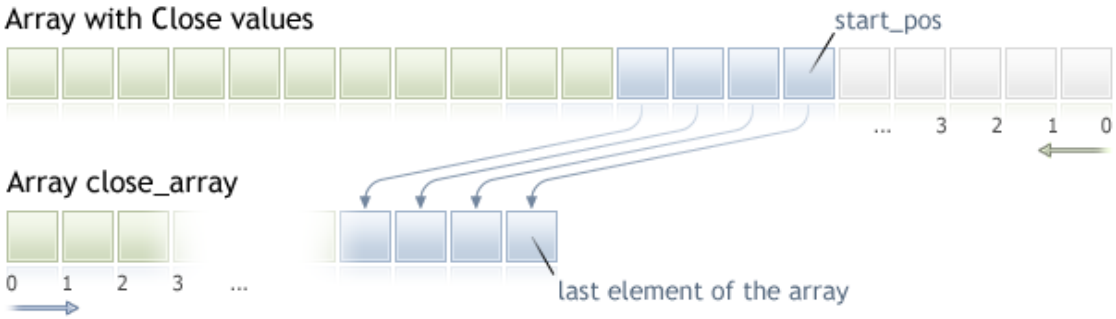
더 보기

CopyHigh

© 2000-2025, MetaQuotes Ltd.

CopyClose

이 함수는 지정된 수량의 선택된 심볼-주기에 대한 막대 종가의 이력 데이터를 close_array에 넣습니다. 요소 순서는 현재에서 과거까지입니다. 즉, 0의 시작 포지션은 현재 막대를 의미합니다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyClose(
   string           symbol_name,       // 심볼명
   ENUM_TIMEFRAMES  timeframe,         // 주기
   int              start_pos,         // 시작 포지션
   int              count,             // 복사할 데이터 카운트
   double           close_array[]      // 복사할 타겟 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyClose(
   string           symbol_name,       // 심볼명
   ENUM_TIMEFRAMES  timeframe,         // 주기
   datetime         start_time,        // 시작 일시
   int              count,             // 복사할 데이터 카운트
   double           close_array[]      // 복사할 타겟 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyClose(
   string           symbol_name,       // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,          // 주기
   datetime          start_time,         // 시작 일시
   datetime          stop_time,          // 정지 일시
   double            close_array[]       // 복사할 타겟 배열
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

start_time

[in] 복사할 첫 번째 요소의 시작 시간.

stop_time

[in] 마지막 복사 요소에 해당하는 막대 시간.

close_array[]

[out] double 유형의 배열.

반환 값

오류 발생 시 복사된 데이터 개수 또는 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면, 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

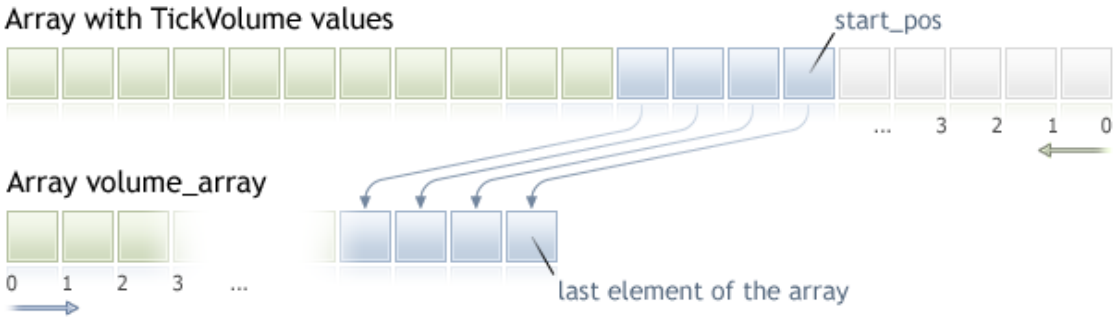
객체 바인딩 방법 섹션에서 요청 기록 데이터에 대한 자세한 예를 참조하십시오. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyTickVolume

이 함수는 지정된 수량의 선택한 심볼-주기에 대한 틱 거래량의 기록 데이터를 volume_array에 가져옵니다. 요소 순서는 현재에서 과거까지입니다. 즉, 0의 시작 포지션은 현재 막대를 의미합니다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyTickVolume(
   string           symbol_name,      // 심볼명
   ENUM_TIMEFRAMES  timeframe,        // 주기
   int              start_pos,        // 포지션 시작
   int              count,            // 복제할 데이터 수
   long             volume_array[]    // 틱 볼륨의 대상 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyTickVolume(
   string           symbol_name,      // 심볼명
   ENUM_TIMEFRAMES  timeframe,        // 주기
   datetime         start_time,       // 시작 일시
   int              count,            // 복제할 데이터 수
   long             volume_array[]    // 틱 볼륨의 대상 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyTickVolume(
   string           symbol_name,      // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,        // 주기
   datetime          start_time,       // 시작 일시
   datetime          stop_time,        // 정지 일시
   long              volume_array[]    // 틱 볼륨의 대상 배열
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

start_time

[in] 복사할 첫 번째 요소의 시작 시간.

stop_time

[in] 마지막 복사 요소에 해당하는 막대 시간.

volume_array[]

[out] long 타입의 배열.

반환 값

오류 발생 시 복사된 데이터 개수 또는 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

예:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot TickVolume
#property indicator_label1  "TickVolume"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  C'143,188,139'
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 입력 매개 변수
input int      bars=3000;
//--- 지표 버퍼
double         TickVolumeBuffer[];
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 지표 버퍼 연결
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      long timeseries[];
```

© 2000-2025, MetaQuotes Ltd.

```
      ArraySetAsSeries(timeseries,true);
      int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
      for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
      for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-
      Print("다음과 같은 수의 TickVolume 값을 받았습니다: "+prices);
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

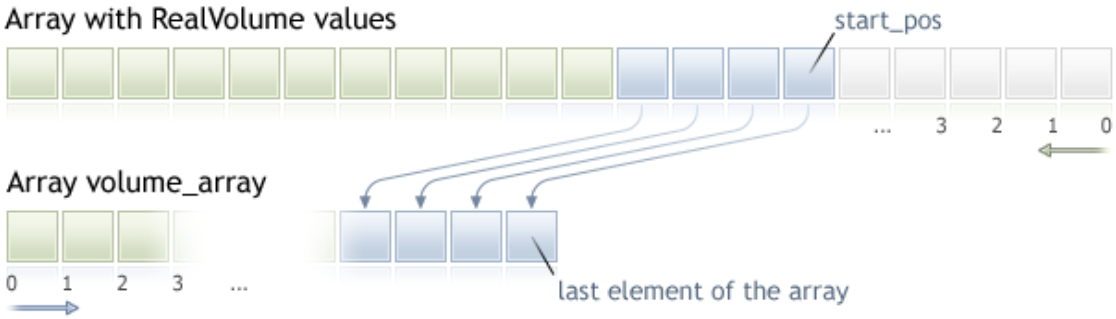
객체 바인딩 방법 섹션에서 요청 기록 데이터에 대한 자세한 예를 참조하십시오. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyRealVolume

이 함수는 지정된 수량의 선택된 심볼-주기에 대한 거래량의 기록 데이터를 volume_array에 가져옵니다. 요소 순서는 현재에서 과거까지입니다. 즉, 0의 시작 포지션은 현재 막대를 의미합니다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopyRealVolume(
   string           symbol_name,      // 심볼명
   ENUM_TIMEFRAMES  timeframe,        // 주기
   int              start_pos,        // 포지션 시작
   int              count,            // 복제할 데이터 수
   long             volume_array[]    // 거래량 값에 대한 대상 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopyRealVolume(
   string           symbol_name,      // 심볼명
   ENUM_TIMEFRAMES  timeframe,        // 주기
   datetime         start_time,       // 시작 일시
   int              count,            // 복제할 데이터 수
   long             volume_array[]    // 거래량 값에 대한 대상 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopyRealVolume(
   string           symbol_name,      // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,        // 주기
   datetime          start_time,       // 시작 일시
   datetime          stop_time,        // 정지 일시
   long              volume_array[]    // 거래량 값에 대한 대상 배열
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

start_time

[in] 복사할 첫 번째 요소의 시작 시간.

stop_time

[in] 마지막 복사 요소에 해당하는 막대 시간.

volume_array[]

[out] long 타입의 배열.

반환 값

오류 발생 시 복사된 데이터 개수 또는 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

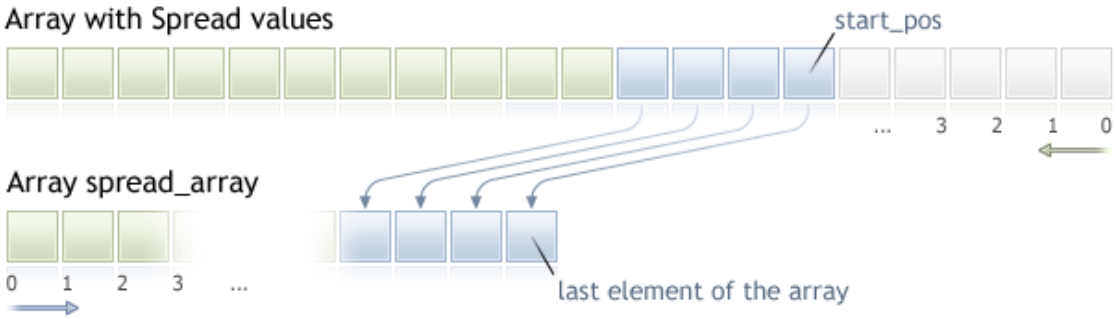
개체 바인딩 방법 섹션에서 요청 기록 데이터의 예를 참조하십시오.. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopySpread

이 함수는 지정된 수량의 선택한 심볼-주기에 대한 확산 값의 기록 데이터를 spread_array에 가져옵니다. 요소 순서는 현재에서 과거까지입니다. 즉, 0의 시작 포지션은 현재 막대를 의미합니다.



아직 알 수 없는 양의 데이터를 복사할 때, 권장하는 것은 동적 배열을 타겟 배열로 사용하는 것이며, 이는 요청된 데이터 수가 대상 어레이의 길이보다 작거나 많을 경우 요청된 데이터가 완전히 맞도록 메모리를 재할당하려고 시도하기 때문입니다.

복사해야 하는 데이터의 양을 알고 있다면, 과도한 메모리를 할당하지 않도록 정적으로 할당된 버퍼에 복사하는 것이 좋습니다.

대상 어레이의 속성(as_series=true 또는 as_series=false)에 관계 없음 가장 오래된 요소가 어레이에 할당된 물리적 메모리의 시작 부분에 위치하도록 데이터가 복사됩니다. 함수 호출에는 세 가지 종류가 있습니다.

첫 번째 포지션과 필요한 요소 수에 따라 호출

```
int  CopySpread(
   string            symbol_name,      // 심볼명
   ENUM_TIMEFRAMES   timeframe,        // 주기
   int               start_pos,        // 포지션 시작
   int               count,            // 복제할 데이터 수
   int               spread_array[]    // 스프레드 값에 대한 대상 배열
   );
```

시작 날짜 및 필요한 요소 수에 따라 호출

```
int  CopySpread(
   string            symbol_name,      // 심볼명
   ENUM_TIMEFRAMES   timeframe,        // 주기
   datetime          start_time,       // 시작 일시
   int               count,            // 복제할 데이터 수
   int               spread_array[]    // 스프레드 값에 대한 대상 배열
   );
```

필요한 시간 간격의 시작 및 종료 날짜로 호출

```
int  CopySpread(
   string            symbol_name,      // 심볼명
```

© 2000-2025, MetaQuotes Ltd.

```
   ENUM_TIMEFRAMES   timeframe,        // 주기
   datetime          start_time,       // 시작 일시
   datetime          stop_time,        // 정지 일시
   int               spread_array[]    // 스프레드 값에 대한 대상 배열
   );
```

Parameters

symbol_name

[in] 심볼명.

timeframe

[in] 주기.

start_pos

[in] 복사할 첫 번째 요소의 시작 위치.

count

[in] 복사할 데이터 수.

start_time

[in] 복사할 첫 번째 요소의 시작 시간.

stop_time

[in] 마지막 복사 요소에 해당하는 막대 시간.

spread_array[]

[out] Array of int type.

반환 값

오류 발생 시 복사된 데이터 개수 또는 -1을 반환합니다.

참고

요청된 데이터의 전체 간격이 서버의 사용 가능한 데이터 범위를 벗어나면 함수는 -1을 반환합니다. TERMINAL_MAXBARS 외부의 데이터(차트의 최대 막대 수)를 요청하면 함수도 -1을 반환합니다.

지표에서 데이터를 요청할 때, 아직 구축되지 않았거나 서버에서 다운받아야 할 경우, 함수는 즉시 -1을 반환하지만 다운로드/구축 과정이 시작됩니다.

Expert Advisor 또는 스크립트에서 데이터를 요청할 때 터미널에 이러한 데이터가 로컬에 없으면 서버에서 다운로드가 시작되거나 로컬 기록에서 데이터를 빌드할 수 있지만 아직 준비되지 않은 경우 필수 시계열의 구축이 시작됩니다. 이 함수는 시간 초과 만료 순간까지 준비될 데이터 양을 반환하지만 기록 다운로드가 계속되며, 다음 유사한 요청에 따라 기능이 더 많은 데이터를 반환합니다.

시작 날짜와 필수 요소 수까지 데이터를 요청할 경우, 날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 값이 반환되는 막대의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

지정된 날짜 범위 내에서 데이터를 요청하면 이 간격의 데이터만 반환됩니다. 간격이 설정되고 초까지 계산됩니다. 즉, 값이 반환되는 바의 오픈 시간(거래량, 스프레드, 지표 버퍼의 값, 시가, 고가, 저가, 종가 또는 오픈 시간)은 항상 요청된 간격 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

따라서, 현재 날짜가 토요일인 경우, *start_time=Last_Tuesday*와 *stop_time=Last_Friday*를 지정하는 주 타임프레임에 데이터를 복사하려고 할 때, 주 타임프레임의 오픈 시간은 항상 일요일이지만, 주 막대는 지정된 간격에 속하지 않기 때문에 함수는 0을 반환합니다.

완료되지 않은 현재 막대에 해당하는 값을 반환해야 하는 경우 *start_pos*=0 및 *count*=1를 지정하는 첫 번째 호출 형식을 사용할 수 있습니다.

예:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Spread
#property indicator_label1  "Spread"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 입력 매개 변수
input int      bars=3000;
//--- 지표 버퍼
double         SpreadBuffer[];
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 지표 버퍼 연결
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      int spread_int[];
```

© 2000-2025, MetaQuotes Ltd.

```
      ArraySetAsSeries(spread_int,true);
      int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
      Print("다음과 같은 수의 스프레드 값을 받았습니다: ",spreads);
      for (int i=0;i<spreads;i++)
      {
      SpreadBuffer[rates_total-1-i]=spread_int[i];
      if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
      }
     }
   else
     {
      double Ask,Bid;
      Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
      Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
      Comment("Ask = ",Ask,"  Bid = ",Bid);
      SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

개체 바인딩 방법 섹션에서 요청 기록 데이터의 예를 참조하십시오.. 이 섹션에서 사용할 수 있는 스크립트에서는 마지막 1000개의 막대에서 지표 iFractals의 값을 가져오는 방법과 마지막 10개의 위쪽 및 아래쪽 프랙탈을 차트에 표시하는 방법을 보여 줍니다. 누락 데이터가 있으며 일반적으로 다음 스타일을 사용하여 그려진 모든 지표에도 유사한 기법을 사용할 수 있습니다:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

© 2000-2025, MetaQuotes Ltd.

CopyTicks

이 함수는 MqlTick ticks_array로의 형태에서 틱을 수신합니다. 이 경우 틱은 과거에서 현재로 인덱싱되며, 즉, 0 인덱스는 배열에서 가장 오래된 틱입니다. 틱 분석을 위해서는 틱에서 정확히 변경된 내용을 보여주는 *플래그* 필드를 확인하십시오.

```
int  CopyTicks(
   string           symbol_name,           // 심볼명
   MqlTick&         ticks_array[],         // 배열을 수신하는 틱
   uint             flags=COPY_TICKS_ALL,  // 수신된 틱 유형을 결정하는 플래그
   ulong            from=0,                // 틱 요청 시작일
   uint             count=0                // 수신할 틱 수
   );
```

Parameters

symbol_name

[in] 심볼.

ticks_array

[out] 틱 수신용 MqlTick 유형의 배열.

flags

[in] 요청된 틱 유형을 정의하는 플래그. COPY_TICKS_INFO – 입찰 및/또는 요청 사항이 있는 틱, COPY_TICKS_TRADE – Last 및 Volume이 변경된 틱, COPY_TICKS_ALL – 모든 틱. 모든 요청 유형의 경우 이전 틱 값이 MqlTick 구조의 나머지 필드에 추가됩니다.

from

[in] 틱 요청 시작일. 1970.01.01 이후 밀리초. *from*=0이면, 최근 *count* 틱이 반환됩니다.

count

[in] 요청된 틱 수. 'from' 및 'count' 매개 변수를 지정하지 않으면 사용 가능한 모든 최근 눈금(2000개 이하)이 ticks_array[]에 기록됩니다.

Returned value

오류가 발생한 경우 복사된 틱의 수 또는 -1입니다.

참고

CopyTicks() 함수를 사용하면 수신된 모든 틱을 요청하고 분석할 수 있습니다. CopyTicks()를 처음 호출하면 하드 디스크에 저장된 심볼의 틱 데이터베이스 동기화가 시작됩니다. 로컬 데이터베이스가 요청된 모든 틱을 제공하지 않는 경우 누락된 틱이 트레이드 서버에서 자동으로 다운로드됩니다. CopyTicks()에 지정된 ***from*** 날짜부터 시작하여 현재 순간이 동기화될 때까지 체크합니다. 그런 다음 이 심볼에 대해 도착하는 모든 틱이 틱 데이터베이스에 추가되므로 동기화된 상태를 유지합니다.

from 및 *count* 매개 변수를 지정하지 않으면 사용 가능한 모든 최근 틱(2000개 이하)이 ***ticks_array[]***에 기록됩니다. *플래그* 매개 변수를 사용하면 필요한 틱 유형을 지정할 수 있습니다.

COPY_TICKS_INFO – 입찰 및/또는 가격 변경이 포함된 틱이 반환됩니다. 다른 필드의 데이터도 추가됩니다. 예를 들어, Bid만 변경된 경우 *요청* 및 *거래량* 필드는 마지막으로 알려진 값으로 채워집니다. 변경된 내용을 정확하게 확인하려면 TICK_FLAG_BID 및/또는 TICK_FLAG_ASK 값을 갖는 *플래그* 필드를 분석합니다. 틱이 입찰 및 청구가격의 값을 0으로 하고 플래그에 해당 데이터가 변경되었음을 나타내는

© 2000-2025, MetaQuotes Ltd.

경우 (*플래그*=TICK_FLAG_BID|TICK_FLAG_ASK), 이것은 주문장(Market Depth)가 비어 있음을 의미합니다. 다시 말해, 사고파는 주문이 없습니다.

COPY_TICKS_TRADE – 마지막 가격 및 거래량 변경 내용이 포함된 틱이 반환됩니다. 다른 필드의 데이터도 추가됩니다. 즉, 마지막으로 알려진 입찰 및 요청 값이 해당 필드에 지정됩니다. 변경된 내용을 정확하게 확인하려면 TICK_FLAG_LAST 및 TICK_FLAG_VOLUME 값을 갖는 *플래그* 필드를 분석합니다.

COPY_TICKS_ALL – 변경 내용이 포함된 모든 틱이 반환됩니다. 변경되지 않은 필드는 마지막으로 알려진 값으로 채워집니다.

COPY_TICKS_ALL 플래그가 있는 CopyTicks()를 호출하면 요청 간격에서 모든 틱이 즉시 반환되는 반면, 다른 모드의 호출에서는 틱을 처리하고 선택하는 데 약간의 시간이 필요하므로 속도 이점이 크지 않습니다.

틱 요청시(COPY_TICKS_INFO 또는 COPY_TICKS_TRADE), 모든 틱에는 틱 시간의 전체 가격 정보(*입찰, 요청, 최근* 및 *거래량*)가 포함됩니다. 이 함수는 각 틱의 시점의 거래 상태를 보다 쉽게 분석할 수 있도록 제공되므로, 상세 틱 내역을 요청하거나 다른 필드의 값을 검색할 필요가 없습니다.

지표에서 CopyTicks() 함수는 다음과 같은 결과를 반환합니다: 지표에서 호출되면 CopyTick()은 즉시 심볼의 사용 가능한 모든 틱을 반환하고 사용 가능한 데이터가 충분하지 않은 경우 틱 데이터베이스의 동기화를 시작합니다. 하나의 심볼에 있는 모든 지표는 하나의 공통된 스레드에서 작동하므로 동기화가 완료될 때까지 기다릴 수 없습니다. 동기화 후 CopyTicks()는 다음 호출 중에 요청된 모든 틱을 반환합니다. 지표에서 OnCalculate() 함수는 각 틱이 도착한 후에 호출됩니다.

CopyTicks()는 Expert Advisor 및 스크립트에서 결과를 45초간 기다릴 수 있습니다: 지표와 달리 모든 Expert Advisor와 스크립트는 별도의 스레드로 작동하므로 동기화가 완료될 때까지 45초 동안 기다릴 수 있습니다. 이 시간 동안 필요한 양의 틱이 동기화되지 않으면 CopyTicks()는 시간 초과 시 사용 가능한 틱을 반환하고 동기화를 계속합니다. OnTick()은 모든 틱을 처리하는 것은 아니며 시장의 변화에 대해서만 Expert Advisor에게 알립니다. 이것은 변경 사항 일괄 처리일 수 있습니다: 터미널에서 몇 가지 틱을 동시에 할 수 있지만 OnTick()은 한 번만 호출되어 EA에 최신 마켓의 상태를 알립니다.

데이터 반환 속도: 고속 액세스 캐시에서 각 금융상품당 4,096개의 마지막 틱(Market Depth가 실행 중인 심볼의 경우 65,536개의 틱)에 터미널이 저장됩니다. 현재 거래 세션에 대해 요청된 틱이 캐시를 벗어나면 CopyTicks()가 터미널 메모리에 저장된 틱을 호출합니다. 이러한 요청은 실행하는 데 더 많은 시간이 필요합니다. 이 경우 디스크에서 데이터를 읽기 때문에 다른 날짜에 대해 틱을 요청하는 요청이 가장 느립니다.

예:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- 전체 틱 기록을 수신하기 위해 1억 개의 틱을 요청합니다
input int      getticks=100000000; // 요청된 틱 수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int     attempts=0;     // 시도 횟수
   bool    success=false;  // 틱 복사에 성공한 플래그
```

© 2000-2025, MetaQuotes Ltd.

```
   MqlTick tick_array[];   // 배열을 수신하는 틱
   MqlTick lasttick;       // 마지막 틱 데이터를 수신
   SymbolInfoTick(_Symbol,lasttick);
//--- 틱 수신을 3회 시도
   while(attempts<3)
     {
      //--- 틱 수신 전 시작 시간 측정
      uint start=GetTickCount();
//--- 1970.01.01 00:00.001 이후의 틱 기록 요청(from=1 ms인 매개 변수)
      int received=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,1,getticks);
      if(received!=-1)
        {
         //--- 틱 수 및 소요 시간에 대한 정보 표시
         PrintFormat("%s: received %d ticks in %d ms",_Symbol,received,GetTickCount()-
         //--- 틱 기록이 동기화되면 오류 코드가 0입니다
         if(GetLastError()==0)
           {
            success=true;
            break;
           }
         else
            PrintFormat("%s: 틱이 아직 동기화되지 않았습니다. %d 개 틱 수신 (%d ms). Error=
            _Symbol,received,GetTickCount()-start,_LastError);
        }
      //--- 시도 횟수 계산
      attempts++;
      //--- 틱 데이터베이스의 동기화가 종료될 때까지 대기하기 위해 1초 일시 중지
      Sleep(1000);
     }
//--- 세 번의 시도 중 틱 기록 시작부터 요청된 틱을 수신하지 못했습니다.
   if(!success)
     {
      PrintFormat("오류! 세 번의 시도 동안 %s 개 중 %d 개 틱 수신 실패",getticks,_Symbol);
      return;
     }
   int ticks=ArraySize(tick_array);
//--- 배열의 첫 번째 틱 시간 표시
   datetime firstticktime=tick_array[ticks-1].time;
   PrintFormat("최근 틱 시간 = %s.%03I64u",
               TimeToString(firstticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar
//--- 배열의 마지막 틱 시간 표시
   datetime lastticktime=tick_array[0].time;
   PrintFormat("첫번째 틱 시간 = %s.%03I64u",
               TimeToString(lastticktime,TIME_DATE|TIME_MINUTES|TIME_SECONDS),tick_ar

//---
   MqlDateTime today;
   datetime current_time=TimeCurrent();
   TimeToStruct(current_time,today);
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("current_time=%s",TimeToString(current_time));
   today.hour=0;
   today.min=0;
   today.sec=0;
   datetime startday=StructToTime(today);
   datetime endday=startday+24*60*60;
   if((ticks=CopyTicksRange(_Symbol,tick_array,COPY_TICKS_ALL,startday*1000,endday*100
     {
      PrintFormat("CopyTicksRange(%s,tick_array,COPY_TICKS_ALL,%s,%s) 실패, 오류 %d",
                  _Symbol,TimeToString(startday),TimeToString(endday),GetLastError());
      return;
     }
   ticks=MathMax(100,ticks);
//--- 마지막 날의 처음 100개 틱 표시
   int counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         counter++;
         PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
        }
     }
//--- 마지막 날의 처음 100개 딜(deal) 표시
   counter=0;
   for(int i=0;i<ticks;i++)
     {
      datetime time=tick_array[i].time;
      if((time>=startday) && (time<endday) && counter<100)
        {
         if(((tick_array[i].flags&TICK_FLAG_BUY)==TICK_FLAG_BUY) || ((tick_array[i].fl
           {
            counter++;
            PrintFormat("%d. %s",counter,GetTickDescription(tick_array[i]));
           }
        }
     }
  }
//+------------------------------------------------------------------+
//| 틱의 문자열 설명을 반환합니다                                    |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03d ",
                            TimeToString(tick.time),tick.time_msc%1000);
//--- 플래그 확인
   bool buy_tick=((tick.flags&TICK_FLAG_BUY)==TICK_FLAG_BUY);
   bool sell_tick=((tick.flags&TICK_FLAG_SELL)==TICK_FLAG_SELL);
```

© 2000-2025, MetaQuotes Ltd.

```
   bool ask_tick=((tick.flags&TICK_FLAG_ASK)==TICK_FLAG_ASK);
   bool bid_tick=((tick.flags&TICK_FLAG_BID)==TICK_FLAG_BID);
   bool last_tick=((tick.flags&TICK_FLAG_LAST)==TICK_FLAG_LAST);
   bool volume_tick=((tick.flags&TICK_FLAG_VOLUME)==TICK_FLAG_VOLUME);
//--- 먼저 틱에서 거래 플래그 확인
   if(buy_tick || sell_tick)
     {
      //--- 트레이딩 체크에 대한 결과 구성
      desc=desc+(buy_tick?StringFormat("Buy Tick: Last=%G Volume=%d ",tick.last,tick.v
      desc=desc+(sell_tick?StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last,tick
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+"(Trade tick)";
     }
   else
     {
      //--- 정보성 틱을 위해 다른 결과 형성
      desc=desc+(ask_tick?StringFormat("Ask=%G ",tick.ask):"");
      desc=desc+(bid_tick?StringFormat("Bid=%G ",tick.ask):"");
      desc=desc+(last_tick?StringFormat("Last=%G ",tick.last):"");
      desc=desc+(volume_tick?StringFormat("Volume=%d ",tick.volume):"");
      desc=desc+"(Info tick)";
     }
//--- 틱 설명 반환
   return desc;
  }
//+------------------------------------------------------------------+
/* 결과 예시
Si-12.16: received 11048387 ticks in 4937 ms
Last tick time = 2016.09.26 18:32:59.775
First tick time = 2015.06.18 09:45:01.000
1.  2016.09.26 09:45.249 Ask=65370 Bid=65370 (Info tick)
2.  2016.09.26 09:47.420 Ask=65370 Bid=65370 (Info tick)
3.  2016.09.26 09:50.893 Ask=65370 Bid=65370 (Info tick)
4.  2016.09.26 09:51.827 Ask=65370 Bid=65370 (Info tick)
5.  2016.09.26 09:53.810 Ask=65370 Bid=65370 (Info tick)
6.  2016.09.26 09:54.491 Ask=65370 Bid=65370 (Info tick)
7.  2016.09.26 09:55.913 Ask=65370 Bid=65370 (Info tick)
8.  2016.09.26 09:59.350 Ask=65370 Bid=65370 (Info tick)
9.  2016.09.26 09:59.678 Bid=65370 (Info tick)
10. 2016.09.26 10:00.000 Sell Tick: Last=65367 Volume=3 (Trade tick)
11. 2016.09.26 10:00.000 Sell Tick: Last=65335 Volume=45 (Trade tick)
12. 2016.09.26 10:00.000 Sell Tick: Last=65334 Volume=95 (Trade tick)
13. 2016.09.26 10:00.191 Sell Tick: Last=65319 Volume=1 (Trade tick)
14. 2016.09.26 10:00.191 Sell Tick: Last=65317 Volume=1 (Trade tick)
15. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=1 (Trade tick)
16. 2016.09.26 10:00.191 Sell Tick: Last=65316 Volume=10 (Trade tick)
17. 2016.09.26 10:00.191 Sell Tick: Last=65315 Volume=5 (Trade tick)
18. 2016.09.26 10:00.191 Sell Tick: Last=65313 Volume=3 (Trade tick)
```

© 2000-2025, MetaQuotes Ltd.

```
19. 2016.09.26 10:00.191 Sell Tick: Last=65307 Volume=25 (Trade tick)
20. 2016.09.26 10:00.191 Sell Tick: Last=65304 Volume=1 (Trade tick)
21. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=1 (Trade tick)
22. 2016.09.26 10:00.191 Sell Tick: Last=65301 Volume=10 (Trade tick)
23. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=5 (Trade tick)
24. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=1 (Trade tick)
25. 2016.09.26 10:00.191 Sell Tick: Last=65300 Volume=6 (Trade tick)
26. 2016.09.26 10:00.191 Sell Tick: Last=65299 Volume=1 (Trade tick)
27. 2016.09.26 10:00.191 Bid=65370 (Info tick)
28. 2016.09.26 10:00.232 Ask=65297 (Info tick)
29. 2016.09.26 10:00.276 Sell Tick: Last=65291 Volume=31 (Trade tick)
30. 2016.09.26 10:00.276 Sell Tick: Last=65290 Volume=1 (Trade tick)
*/
```

더 보기

SymbolInfoTick, Structure for Current Prices, OnTick()

© 2000-2025, MetaQuotes Ltd.

CopyTicksRange

이 함수는 지정된 날짜 범위 내에서 MqlTick 형식의 틱을 수신하여 ticks_array를 표시합니다. 인덱스는 과거에서 현재로 이동하며 인덱스가 0인 틱이 어레이에서 가장 오래된 틱임을 의미합니다. 틱 분석을 위해 정확히 변경된 내용을 보여주는 *플래그* 필드를 확인하십시오.

```
int  CopyTicksRange(
   const string     symbol_name,           // 심볼명
   MqlTick&         ticks_array[],         // 틱 수신 배열
   uint             flags=COPY_TICKS_ALL,  // 수신되는 틱 유형을 정의하는 플래그
   ulong            from_msc=0,            // 틱 요청 시작일
   ulong            to_msc=0               // 틱 요청 만료일
   );
```

Parameters

symbol_name

[in] 심볼.

ticks_array

[out] 틱 수신용 MqlTick 정적 또는 동적 배열. 정적 배열이 요청된 시간 간격의 모든 틱을 고정할 수 없는 경우 가능한 최대 틱 수가 수신됩니다. 이 경우 함수는 ERR_HISTORY_SMALL_BUFFER (4407) 오류를 생성합니다.

flags

[in] 요청된 틱 유형을 정의하는 플래그. COPY_TICKS_INFO – 입찰 및/또는 요청 사항이 있는 틱, COPY_TICKS_TRADE – Last 및 Volume이 변경된 틱, COPY_TICKS_ALL – 모든 틱. 모든 요청 유형의 경우 이전 틱 값이 MqlTick 구조의 나머지 필드에 추가됩니다.

from_msc

[in] 틱 요청 시작 희망일 1970.01.01 이후 밀리초. *from_msc* 파라미터를 지정하지 않으면 기록 시작 부분의 틱이 전송됩니다. time >= from_msc인 틱이 전송되었습니다.

to_msc

[in] 틱 요청 만료 희망일 밀리초 단위로 1970년 1월 1일 이후 time <= to_msc인 틱이 전송되었습니다. *to_msc* 파라미터를 지정하지 않으면 기록 끝까지 모든 틱이 전송됩니다.

반환 값

오류가 발생한 경우 복사된 틱의 수 또는 -1입니다. GetLastError()는 다음 오류를 반환할 수 있습니다:

- ERR_HISTORY_TIMEOUT – 틱 동기화 대기 시간이 초과되었습니다. 함수가 있는 모든 시간을 전송했습니다.
- ERR_HISTORY_SMALL_BUFFER – 정적 버퍼가 너무 작습니다. 배열을 저장할 수 있는 양만 전송되었습니다.
- ERR_NOT_ENOUGH_MEMORY – 지정된 범위에서 동적 틱 배열로 기록을 수신하기 위한 메모리가 부족합니다.. 틱 배열에 충분한 메모리를 할당하지 못했습니다.

참고

CopyTicksRange() 함수는 지정된 범위(예: 기록의 특정 날짜)에서 엄격하게 틱을 요청하는 데 사용됩니다. 동시에, CopyTicks()은 예를 들어 시작 날짜만 지정할 수 있습니다 – 월 시작부터 현재 순간까지 모든 틱을 수신합니다.

© 2000-2025, MetaQuotes Ltd.

더 보기

SymbolInfoTick, Structure for Current Prices, OnTick, CopyTicks

© 2000–2025, MetaQuotes Ltd.

iBars

내역에서 사용할 수 있는 해당 심볼 및 기간의 막대 수를 반환합니다.

```
int  iBars(
   const string           symbol,          // 심볼
   ENUM_TIMEFRAMES        timeframe        // 주기
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0 은 현재 차트 주기를 의미합니다.

반환 값

해당 심볼 및 기간의 막대 수(내역에서 사용 가능)로 플랫폼 설정에서 "차트의 최대 막대" 매개 변수에서 허용됩니다.

예:

```
  Print("Bar count on the 'EURUSD,H1' is ",iBars("EURUSD",PERIOD_H1));
```

더 보기

Bars

© 2000-2025, MetaQuotes Ltd.

iBarShift

시간별 막대 검색. 이 함수는 지정된 시간에 해당하는 막대 인덱스를 반환합니다.

```
int  iBarShift(
   const string        symbol,          // 심볼
   ENUM_TIMEFRAMES     timeframe,       // 주기
   datetime            time,            // 시간
   bool                exact=false      // 모드
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. PERIOD_CURRENT은 현재 차트를 의미합니다.

time

[in] 검색할 시간 값.

exact=false

[in] 지정된 시간의 막대를 찾을 수 없는 경우 반환 값. *exact=false*인 경우, *iBarShift*는 가장 가까운 막대의 색인을 반환하며, 이 색인의 오픈 시간은 지정된 시간(*time_open<time*)보다 작습니다. 이러한 막대를 찾을 수 없으면(지정된 시간 이전의 내역) 함수는 -1을 반환합니다. *exact=true*인 경우, iBarShift는 가장 가까운 막대를 검색하지 않고 즉시 -1을 반환합니다.

반환 값

지정된 시간에 해당하는 막대의 색인. 지정된 시간에 해당하는 막대가 발견되지 않으면(내역에 공백이 있을 경우), 함수는 -1 또는 가장 가까운 막대의 색인을 반환합니다(*'정확한'*파라미터에 따라 다름).

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 날짜는 일요일입니다.
   datetime time=D'2002.04.25 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- 지정된 시간에 막대가 없으면 iBarShift가 가장 가까운 막대의 색인을 반환합니다.
   int bar_index=iBarShift(symbol,tf,time,exact);
//--- iBarShift() 호출 후 오류 코드를 확인하십시오.
   int error=GetLastError();
   if(error!=0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("iBarShift(): GetLastError=%d - The requested date %s "+
                  "for %s %s는 가용한 내역에서 찾을 수 없습니다",
                  error,TimeToString(time),symbol,EnumToString(tf));
      return;
     }
//--- iBarShift() 기능이 성공적으로 실행되었습니다. exact=false에 대한 결과를 반환합니다.
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),
               DayOfWeek(time),bar_index,string(exact));
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//--- 지정된 시간으로 막대 색인을 요청합니다; 막대가 없으면 -1이 반환됩니다.
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
//--- iBarShift() 기능이 성공적으로 실행되었으며, exact=true에 대한 결과를 반환합니다.
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time)
               ,DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//| 주의 요일 이름을 반환합니다                                      |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
   return day;
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
/* Execution result
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1 (exact=true)
*/
```

© 2000-2025, MetaQuotes Ltd.

iClose

해당 차트에서 막대의 종가('shift' 매개 변수로 표시됨)를 반환합니다.

```
double  iClose(
   const string        symbol,          // 심볼
   ENUM_TIMEFRAMES     timeframe,       // 주기
   int                 shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 종가('shift' 매개 변수로 표시됨) 또는 오류 발생 시 0. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
input int shift=0;
//+------------------------------------------------------------------+
//| 함수-이벤트 핸들러 "틱"                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Open: "  ,DoubleToString(open,Digits()),"\n",
            "High: "  ,DoubleToString(high,Digits()),"\n",
            "Low: "   ,DoubleToString(low,Digits()),"\n",
            "Close: " ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "  ,IntegerToString(bars),"\n"
            );
  }
```

더 보기

CopyClose, CopyRates

© 2000-2025, MetaQuotes Ltd.

iHigh

해당 차트에서 막대의 고가('shift' 매개 변수로 표시됨)을 반환합니다.

```
double  iHigh(
   const string         symbol,          // 심볼
   ENUM_TIMEFRAMES      timeframe,       // 주기
   int                  shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 고가('shift' 매개 변수로 표시됨) 또는 오류 발생 시 0. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
input int shift=0;
//+------------------------------------------------------------------+
//| 함수-이벤트 핸들러 "틱"                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Open: "  ,DoubleToString(open,Digits()),"\n",
            "High: "  ,DoubleToString(high,Digits()),"\n",
            "Low: "   ,DoubleToString(low,Digits()),"\n",
            "Close: " ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "  ,IntegerToString(bars),"\n"
            );
  }
```

더 보기

CopyHigh, CopyRates

© 2000-2025, MetaQuotes Ltd.

iHighest

해당 차트에서 발견된 가장 높은 값의 색인을 반환합니다(현재 막대를 기준으로 이동).

```
int  iHighest(
   const string        symbol,              // 심볼
   ENUM_TIMEFRAMES     timeframe,           // 주기
   ENUM_SERIESMODE     type,                // 시계열 식별자
   int                 count=WHOLE_ARRAY,   // 요소 수
   int                 start=0              // 인덱스
  );
```

Parameters

symbol

[in] 검색을 수행할 심볼. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

type

[in] 검색이 수행될 시계열의 식별자. ENUM_SERIESMODE의 어떤 값과도 같을 수 있습니다.

count=WHOLE_ARRAY

[in] 검색을 수행해야 하는 시계열의 요소 수(현재 막대에서 인덱스 증가 방향으로)

start=0

[in] 가장 높은 값에 대한 검색이 시작되는 초기 막대의 인덱스(현재 막대를 기준으로 이동). 음수 값은 무시되고 0 값으로 대체됩니다.

반환 값

해당 차트에서 발견된 가장 높은 값의 색인(현재 막대를 기준으로 이동) 또는 오류 발생 시 -1. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

예:

```
   double val;
//--- 20개 연속 막대 중 가장 높은 종가 계산
//--- 현재 기간에 인덱스 4에서 인덱스 23까지
   int val_index=iHighest(NULL,0,MODE_CLOSE,20,4);
   if(val_index!=-1)
      val=High[val_index];
   else
      PrintFormat("iHighest() 호출 오류. Error code=%d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

iLow

해당 차트에서 막대의 저가('shift' 매개 변수로 표시됨)를 반환합니다.

```
double  iLow(
   const string         symbol,          // 심볼
   ENUM_TIMEFRAMES      timeframe,       // 주기
   int                  shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 저가('shift' 매개 변수로 표시됨) 또는 오류 발생 시 0. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
input int shift=0;
//+------------------------------------------------------------------+
//| 함수-이벤트 핸들러 "틱"                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Open: "  ,DoubleToString(open,Digits()),"\n",
            "High: "  ,DoubleToString(high,Digits()),"\n",
            "Low: "   ,DoubleToString(low,Digits()),"\n",
            "Close: " ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "  ,IntegerToString(bars),"\n"
            );
  }
```

더 보기

CopyLow, CopyRates

© 2000-2025, MetaQuotes Ltd.

iLowest

해당 차트에서 발견된 가장 작은 값의 색인을 반환합니다(현재 막대를 기준으로 이동).

```
int  iLowest(
   const string        symbol,              // 심볼
   ENUM_TIMEFRAMES     timeframe,           // 주기
   ENUM_SERIESMODE     type,                // 시계열 식별자
   int                 count=WHOLE_ARRAY,   // 요소 수
   int                 start=0              // 인덱스
  );
```

Parameters

symbol

[in] 검색을 수행할 심볼. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

type

[in] 검색이 수행될 시계열의 식별자. ENUM_SERIESMODE의 어떤 값과도 같을 수 있습니다.

count=WHOLE_ARRAY

[in] 검색을 수행해야 하는 시계열의 요소 수(현재 막대에서 인덱스 증가 방향으로)

start=0

[in] 가장 낮은 값에 대한 검색이 시작되는 초기 막대의 인덱스(현재 막대를 기준으로 이동). 음수 값은 무시되고 0 값으로 대체됩니다.

반환 값

해당 차트에서 발견된 가장 낮은 값의 색인(현재 막대를 기준으로 이동) 또는 오류 발생 시 -1. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

예:

```
   double val;
//--- 15개 연속 막대 중 실제 거래량 값이 가장 낮은 막대를 검색
//--- 현재 타임프레임에 인덱스 10부터 인덱스 24까지
   int val_index=iLowest(NULL,0,MODE_REAL_VOLUME,15,10);
   if(val_index!=-1)
      val=Low[val_index];
   else
      PrintFormat("iLowest() call error. Error code=%d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

iOpen

해당 차트에서 막대 시가('shift' 매개 변수로 표시됨)을 반환합니다.

```
double  iOpen(
   const string         symbol,          // 심볼
   ENUM_TIMEFRAMES      timeframe,       // 주기
   int                  shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 시가('shift' 매개 변수로 표시됨) 또는 오류 발생 시 0. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
input int shift=0;
//+------------------------------------------------------------------+
//| 함수-이벤트 핸들러 "틱"                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Open: "  ,DoubleToString(open,Digits()),"\n",
            "High: "  ,DoubleToString(high,Digits()),"\n",
            "Low: "   ,DoubleToString(low,Digits()),"\n",
            "Close: " ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "  ,IntegerToString(bars),"\n"
            );
  }
```

더 보기

CopyOpen, CopyRates

© 2000-2025, MetaQuotes Ltd.

iTime

해당 차트에서 막대의 오픈 시간('shift' 매개 변수로 표시됨)을 반환합니다.

```
datetime  iTime(
   const string        symbol,          // 심볼
   ENUM_TIMEFRAMES     timeframe,       // 주기
   int                 shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 오픈 시간('shift' 매개 변수로 표시됨) 또는 오류 발생 시 0. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 날짜는 일요일입니다.
   datetime time=D'2018.06.10 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- 지정된 시간에 막대가 없습니다. iBarShift가 가장 가까운 막대의 색인을 반환합니다.
   int bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
```

© 2000-2025, MetaQuotes Ltd.

```
                bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//PrintFormat(iTime(symbol,tf,bar_index));
//--- 지정된 시간으로 막대 색인을 요청하지만 막대가 없으므로 -1을 반환합니다
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
  }
//+------------------------------------------------------------------+
//| 주의 요일 이름을 반환합니다                                      |
//+------------------------------------------------------------------+
string DayOfWeek(const datetime time)
  {
   MqlDateTime dt;
   string day="";
   TimeToStruct(time,dt);
   switch(dt.day_of_week)
     {
      case 0: day=EnumToString(SUNDAY);
      break;
      case 1: day=EnumToString(MONDAY);
      break;
      case 2: day=EnumToString(TUESDAY);
      break;
      case 3: day=EnumToString(WEDNESDAY);
      break;
      case 4: day=EnumToString(THURSDAY);
      break;
      case 5: day=EnumToString(FRIDAY);
      break;
      default:day=EnumToString(SATURDAY);
      break;
     }
//---
   return day;
  }
/* The result:
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index is 64 (exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar index is -1 (exact=true)
*/
```

더 보기

CopyTime, CopyRates

© 2000-2025, MetaQuotes Ltd.

iTickVolume

해당 차트에서 막대의 틱 볼륨('shift' 매개 변수로 표시됨)을 반환합니다.

```
long  iTickVolume(
   const string         symbol,          // 심볼
   ENUM_TIMEFRAMES      timeframe,       // 주기
   int                  shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 틱 볼륨('shift' 매개 변수로 표시됨) 또는 오류 발생 시 0입니다. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
input int shift=0;
//+------------------------------------------------------------------+
//| 함수-이벤트 핸들러 "틱"                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Open: "  ,DoubleToString(open,Digits()),"\n",
            "High: "  ,DoubleToString(high,Digits()),"\n",
            "Low: "   ,DoubleToString(low,Digits()),"\n",
            "Close: " ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "  ,IntegerToString(bars),"\n"
            );
  }
```

더 보기

CopyTickVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iRealVolume

해당 차트에서 막대의 실제 거래량('shift' 파라미터로 표시됨)을 반환합니다.

```
long  iRealVolume(
   const string         symbol,          // 심볼
   ENUM_TIMEFRAMES      timeframe,       // 주기
   int                  shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 실제 볼륨('shift' 파라미터로 표시됨) 또는 오류 발생 시 0. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
input int shift=0;
//+------------------------------------------------------------------+
//| 함수-이벤트 핸들러 "틱"                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Open: "  ,DoubleToString(open,Digits()),"\n",
            "High: "  ,DoubleToString(high,Digits()),"\n",
            "Low: "   ,DoubleToString(low,Digits()),"\n",
            "Close: " ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "  ,IntegerToString(bars),"\n"
            );
  }
```

더 보기

CopyRealVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iVolume

해당 차트에서 막대의 틱 볼륨('shift' 매개 변수로 표시됨)을 반환합니다.

```
long  iVolume(
   const string         symbol,          // 심볼
   ENUM_TIMEFRAMES      timeframe,       // 주기
   int                  shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 틱 볼륨('shift' 매개 변수로 표시됨) 또는 오류 발생 시 0입니다. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
input int shift=0;
//+------------------------------------------------------------------+
//| 함수-이벤트 핸들러 "틱"                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Open: "  ,DoubleToString(open,Digits()),"\n",
            "High: "  ,DoubleToString(high,Digits()),"\n",
            "Low: "   ,DoubleToString(low,Digits()),"\n",
            "Close: " ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "  ,IntegerToString(bars),"\n"
            );
  }
```

더 보기

CopyTickVolume, CopyRates

© 2000-2025, MetaQuotes Ltd.

iSpread

해당 차트에서 막대의 스프레드 값('shift' 매개 변수로 표시됨)을 반환합니다.

```
long  iSpread(
   const string         symbol,          // 심볼
   ENUM_TIMEFRAMES      timeframe,       // 주기
   int                  shift            // 시프트
   );
```

Parameters

symbol

[in] 금융상품의 심볼명. NULL은 현재 심볼을 의미합니다.

timeframe

[in] 주기. 이 값은 ENUM_TIMEFRAMES 열거값 중 하나일 수 있습니다. 0은 현재 차트 주기를 의미합니다.

shift

[in] 시계열에서 수신된 값의 인덱스(현재 막대를 기준으로 지정된 막대 수만큼 후진 이동).

반환 값

해당 차트에서 막대의 스프레드 값('shift' 파라미터로 표시됨) 또는 오류 발생 시 0입니다. 오류 세부사항에 대해선, GetLastError() 함수를 호출하십시오.

참고

이 함수는 항상 실제 데이터를 반환합니다. 이를 위해 각 통화 중에 지정된 심볼/주기에 대한 시계열 요청을 수행합니다. 즉, 첫 번째 기능 호출 중에 준비된 데이터가 없을 경우 결과를 준비하는 데 시간이 걸릴 수 있습니다.

이 함수는 이전 호출 결과를 저장하지 않으며 빠른 값 반환을 위한 로컬 캐시가 없습니다.

예:

```
input int shift=0;
//+------------------------------------------------------------------+
//| 함수-이벤트 핸들러 "틱"                                          |
//+------------------------------------------------------------------+
void OnTick()
  {
   datetime time  = iTime(Symbol(),Period(),shift);
   double   open  = iOpen(Symbol(),Period(),shift);
   double   high  = iHigh(Symbol(),Period(),shift);
   double   low   = iLow(Symbol(),Period(),shift);
   double   close = iClose(NULL,PERIOD_CURRENT,shift);
   long     volume= iVolume(Symbol(),0,shift);
   int      bars  = iBars(NULL,0);

   Comment(Symbol(),",",EnumToString(Period()),"\n",
           "Time: "  ,TimeToString(time,TIME_DATE|TIME_SECONDS),"\n",
```

© 2000-2025, MetaQuotes Ltd.

```
            "Open: "  ,DoubleToString(open,Digits()),"\n",
            "High: "  ,DoubleToString(high,Digits()),"\n",
            "Low: "   ,DoubleToString(low,Digits()),"\n",
            "Close: " ,DoubleToString(close,Digits()),"\n",
            "Volume: ",IntegerToString(volume),"\n",
            "Bars: "  ,IntegerToString(bars),"\n"
            );
  }
```

더 보기

CopySpread, CopyRates

© 2000-2025, MetaQuotes Ltd.

Custom symbols

사용자 지정 심볼 속성을 생성하고 편집하는 기능.

특정 거래 서버 터미널에 연결하면 브로커가 제공하는 재무 기호의 시계열 작업을 할 수 있습니다. 사용 가능한 재무 기호가 Market Watch(마켓 워치) 창에 목록으로 표시됩니다. 별도의 기능 그룹을 통해 거래 세션, 시장 수준 업데이트 및심볼 속성에 대한 데이터를 수신할 수 있습니다.

이 섹션에서 설명하는 기능 그룹을 통해 사용자 지정 심볼을 생성할 수 있습니다. 이를 위해 사용자는 거래 서버의 기존 심볼, 텍스트 파일 또는 외부 데이터 소스를 적용할 수 있습니다.

기능	액션
CustomSymbolCreate	지정된 그룹에 지정된 이름을 가진 사용자 지정 심볼을 생성
CustomSymbolDelete	지정된 이름의 사용자 지정 심볼 삭제
CustomSymbolSetInteger	사용자 지정 심볼에 대한 정수 유형 속성 값 설정
CustomSymbolSetDouble	사용자 지정 심볼에 대한 실제 유형 속성 값 설정
CustomSymbolSetString	사용자 지정 심볼에 대한 문자열 유형 속성 값 설정
CustomSymbolSetMarginRate	사용자 지정 심볼에 대한 주문 유형 및 방향에 따른 마진율 설정
CustomSymbolSetSessionQuote	지정된 심볼 및 평일에 대해 지정된 시세 세션의 시작 및 종료 시간 설정
CustomSymbolSetSessionTrade	지정된 심볼 및 평일에 대해 지정된 트레이딩 세션의 시작 및 종료 시간 설정
CustomRatesDelete	지정된 시간 간격의 사용자 지정 심볼 가격 기록에서 모든 막대 삭제
CustomRatesReplace	지정된 시간 간격 내에 사용자 지정 심볼의 가격 기록을 MqlRates 유형 배열의 대열로 완전히 교체
CustomRatesUpdate	사용자 지정 심볼 기록에 결손 막대를 추가하고 기존 데이터를 MqlRates 유형 배열의 막대로 교체
CustomTicksAdd	MqlTick 유형의 배열에서 사용자 지정 심볼의 가격 기록에 데이터를 추가 Market Watch 창에서 사용자 지정 심볼을 선택해야 함
CustomTicksDelete	지정된 시간 간격의 사용자 지정 심볼 가격 기록에서 모든 틱 삭제
CustomTicksReplace	지정된 시간 간격 내에 사용자 지정 심볼의 가격 기록을 MqlTick 유형 배열의 데이터로 완전히 교체
CustomBookAdd	다음을 전달 – 사용자 지정 심볼에 대한 마켓의 뎁스(Depth of Market)의 상태

© 2000-2025, MetaQuotes Ltd.

CustomSymbolCreate

지정된 그룹에 지정된 이름의 사용자 지정 심볼을 생성.

```
bool  CustomSymbolCreate(
   const string      symbol_name,          // 사용자 지정 심볼 이름
   const string      symbol_path="",       // 심볼을 생성할 그룹 이름
   const string      symbol_origin=NULL    // 사용자 지정 심볼을 생성하기 위한 기준으로 사용도
   );
```

매개변수

symbol_name

[in] 사용자 지정 심볼 이름. 심볼이 위치한 그룹 또는 하위 그룹을 포함해서는 안됩니다.

symbol_path=""

[in] 심볼이 있는 그룹 이름

symbol_origin=NULL

[in] 생성된 사용자 지정 심볼의 속성을 복사할 심볼 이름. 사용자 지정 심볼을 생성한 후 적절한 기능을 사용하여 속성 값을 필요한 속성 값으로 변경할 수 있습니다.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

모든 사용자 지정 심볼은 특수 사용자 지정 섹션에 생성됩니다. 그룹 이름이 지정되지 않은 경우 (사용자 지정 심볼 생성 기능의 *symbol_path* 매개 변수에 빈 문자열 또는 NULL 포함), 사용자 지정 섹션 루트에 사용자 지정 심볼이 생성됩니다. 여기서는 파일 시스템과 유사할 수 있으며, 여기서 그룹 및 하위 폴더를 폴더 및 하위 폴더로 볼 수 있습니다

심볼 및 그룹 이름에는 문장 부호, 공백 또는 특수 문자(".", "_", "&" 및 "#"만 포함할 수 있음)가 없는 라틴 문자만 포함할 수 있습니다. <, >, :, ", /, \, |, ?, * 문자는 사용하지 않는 것이 좋습니다.

사용자 지정 심볼 이름은 생성된 그룹 이름에 관계 없이 고유해야 합니다. 동일한 이름의 심볼이 이미 있는 경우 CustomSymbolCreate() 함수는 'false'를 반환하고 이후 GetLastError() 호출에서는 오류 5300 (ERR_NOT_CUSTOM_SYMBOL) 또는 5304 (ERR_CUSTOM_SYMBOL_EXIST)를 반환합니다.

심볼 이름의 길이는 31자를 초과할 수 없습니다. 그렇지 않으면, CustomSymbolCreate()가 'false'를 반환하고 오류 5302 - ERR_CUSTOM_SYMBOL_NAME_LONG이 활성화됩니다.

symbol_path 매개변수는 두 가지 방법으로 설정할 수 있습니다:

- 그룹 이름만 사용할 수 있습니다(예: "CFD₩Metals"). 오류를 방지하려면 이 옵션을 사용하는 것이 가장 좋습니다.
- 또는 <그룹> 이름 + 그룹 구분 기호 "₩"+<사용자 지정 심볼 이름>, 예 - "CFD₩Metals₩Platinum". 이 경우 그룹 이름은 사용자 지정 기호의 정확한 이름으로 끝나야 합니다. 일치하지 않는 경우 사용자 지정 기호가 계속 생성되지만 원하는 그룹에는 생성되지 않습니다. 예를 들어, *symbol_path*="CFD\\Metals\\Platinum" and *symbol_name*="platinum" (register error), 뒤에 "Custom₩CFD₩Metals₩Platinum" 그룹에 사용자 지정 심볼 이름 "platinum" 이 생성됩니다. SymbolInfoGetString("platinum",SYMBOL_PATH) 함수는 "Custom₩CFD₩Metals₩Platinum₩platinum" 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SYMBOL_PATH 속성은 끝에 심볼 이름이 있는 경로를 반환합니다. 따라서 동일한 그룹에 사용자 지정 심볼을 생성하려는 경우 변경 없이 복사할 수 없습니다. 이 경우 위에서 설명한 결과를 얻지 않으려면 심볼 이름을 잘라야 합니다.

존재하지 않는 기호를 *symbol_origin* 매개변수로 설정하면 *symbol_origin* 매개변수가 설정되지 않은 것처럼 사용자 지정 심볼이 빈 상태로 생성됩니다. 이 경우 오류 4301 - ERR_MARKET_UNKNOWN_SYMBOL 이 활성화됩니다.

symbol_path 매개변수 길이는 "Custom₩", "₩그룹 구분 심볼 및 심볼 이름(끝에 지정된 경우)을 고려할 때 127자를 초과하지 않아야 합니다.

예:

```
//+------------------------------------------------------------------+
//|                                          CustomSymbolCreate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // 사용자 지정 심볼 명
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 심볼이 생성될 그룹의 이름
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // 사용자 정의 심볼의 기반이 되는 심볼 명

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성하지 못한 경우 저널에 이를 알립니다.
   if(!CustomSymbolCreate(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL_ORIGIN
     {
      Print("CustomSymbolCreate() failed. Error ", GetLastError());
      return;
     }

//--- 생성된 심볼이 있는지 확인하고 생성된 그룹을 가져옵니다.in
   bool   custom= false;
   bool   exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   string path  = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_PATH);

//--- 심볼 생성 결과와 그룹 이름(심볼 속성에서 지정하고 얻은)을 저널에 인쇄합니다.
   PrintFormat("Custom symbol '%s' created\n"+
               "Symbol '%s' is exist: %s\n"+
               "Symbol '%s' is custom: %s\n"+
               "Path specified in the settings: '%s'\n"+
               "Path returned from the 'SYMBOL_PATH' property: '%s'",
```

© 2000-2025, MetaQuotes Ltd.

```
                CUSTOM_SYMBOL_NAME,
                CUSTOM_SYMBOL_NAME, (string)exist,
                CUSTOM_SYMBOL_NAME, (string)custom,
                CUSTOM_SYMBOL_PATH,
                path);

//--- 2초 동안 기다린 후 저널의 결과 메시지와 함께 생성된 심볼을 삭제합니다.
   Sleep(2000);
   ResetLastError();
   bool deleted = CustomSymbolDelete(CUSTOM_SYMBOL_NAME);
   Print(deleted ? StringFormat("Custom symbol '%s' removed", CUSTOM_SYMBOL_NAME) : St
   /*
   결과:
   Custom symbol 'EURUSD.C' created
   Symbol 'EURUSD.C' is exist: true
   Symbol 'EURUSD.C' is custom: true
   Path specified in the settings: 'Forex'
   Path returned from the 'SYMBOL_PATH' property: 'Custom\Forex\EURUSD.C'
   Custom symbol 'EURUSD.C' removed
   */
  }
```

추가 참조

SymbolName, SymbolSelect, CustomSymbolDelete

© 2000-2025, MetaQuotes Ltd.

CustomSymbolDelete

지정한 이름의 사용자 지정 심볼을 삭제.

```
bool  CustomSymbolDelete(
   const string      symbol_name          // 사용자 지정 심볼 이름
   );
```

매개변수

심볼

[in] 사용자 지정 심볼 이름. 기존 심볼의 이름과 일치하지 않아야 합니다.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

Market Watch(마켓 워치)에 표시된 사용자 지정 심볼 또는 차트를 연 심볼을 삭제할 수 없습니다.

예:

```
//+------------------------------------------------------------------+
//|                                          CustomSymbolDelete.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // 사용자 지정 심볼 명
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 심볼이 생성될 그룹의 이름
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // 사용자 정의 심볼의 기반이 되는 심볼 명

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성하지 못한 경우 저널에 이를 알립니다.
   if(!CustomSymbolCreate(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL_ORIGIN
     {
      Print("CustomSymbolCreate() failed. Error ", GetLastError());
      return;
     }

//--- 생성된 심볼이 있는지 확인하고 그 결과를 저널에 출력합니다.
   bool custom= false;
```

© 2000-2025, MetaQuotes Ltd.

```
   bool exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   PrintFormat("Custom symbol '%s' exists: %s", CUSTOM_SYMBOL_NAME, (string)exist);

//--- 2초 동안 기다린 후 저널의 결과 메시지와 함께 생성된 심볼을 삭제합니다.
   Sleep(2000);
   ResetLastError();
   bool deleted = CustomSymbolDelete(CUSTOM_SYMBOL_NAME);
   Print(deleted ? StringFormat("Custom symbol '%s' removed", CUSTOM_SYMBOL_NAME) : St

//--- 생성된 심볼이 있는지 확인하고 그 결과를 저널에 출력합니다.
   exist = SymbolExist(CUSTOM_SYMBOL_NAME, custom);
   PrintFormat("Custom symbol '%s' exists: %s", CUSTOM_SYMBOL_NAME, (string)exist);
   /*
   결과:
   Custom symbol 'EURUSD.C' exists: true
   Custom symbol 'EURUSD.C' removed
   Custom symbol 'EURUSD.C' exists: false
   */
  }
```

추가 참조

SymbolName, SymbolSelect, CustomSymbolCreate

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetInteger

사용자 정의 심볼의 정수 유형 속성 값을 설정.

```
bool  CustomSymbolSetInteger(
   const string               symbol_name,      // 심볼 이름
   ENUM_SYMBOL_INFO_INTEGER   property_id,      // 속성 ID
   long                       property_value    // 속성 값
   );
```

매개변수

symbol_name

[in] 사용자 지정 심볼 이름.

property_id

[in] 심볼 속성 ID. 값은 ENUM_SYMBOL_INFO_INTEGER 열거 값 중 하나일 수 있습니다.

property_value

[in] 속성 값을 포함하는 long 유형 변수.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

심볼 사양에서 다음 속성 중 하나가 변경되면 사용자 지정 심볼의 분 및 틱 내역이 완전히 제거됩니다.

- SYMBOL_CHART_MODE – 막대를 구성하는 가격 유형(입찰 또는 마지막)
- SYMBOL_DIGITS – 가격을 표시할 소수점 뒤의 자릿수

사용자 지정 심볼 내역을 삭제한 후 터미널은 업데이트된 속성을 사용하여 새 내역을 생성하려고 시도합니다. 사용자 지정 심볼 속성을 수동으로 변경할 때도 마찬가지입니다.

예:

```
//+------------------------------------------------------------------+
//|                                      CustomSymbolSetInteger.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // 사용자 지정 심볼 명
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 심볼이 생성될 그룹의 이름
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // 사용자 정의 심볼의 기반이 되는 심볼 명

//+------------------------------------------------------------------+
//| Script program start function                                    |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 사용자 정의 심볼의 기반이 되는 심볼의 속성을 저널에서 가져오고 출력합니다.
//--- (거래 모드, 스탑 주문 설치 수준 및 트레이딩 운영 중지 거리)
   ENUM_SYMBOL_TRADE_EXECUTION origin_exe_mode = (ENUM_SYMBOL_TRADE_EXECUTION)SymbolIn
   int origin_stops_level = (int)SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN, SYMBOL_TRADE_
   int origin_freeze_level= (int)SymbolInfoInteger(CUSTOM_SYMBOL_ORIGIN, SYMBOL_TRADE_

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Deal execution mode: %s\n  Stops Level: %d\n  Freeze Level: %d",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               StringSubstr(EnumToString(origin_exe_mode), 23), origin_stops_level, or

//--- 사용자 정의 심볼 속성에 대한 다른 값을 설정합니다.
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_EXEMODE, SYMBOL_TRADE
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_STOPS_LEVEL, 10);
   res &=CustomSymbolSetInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_FREEZE_LEVEL, 3);

//--- 속성을 설정할 때 오류가 발생하면 저널에 해당 메시지를 표시합니다.
   if(!res)
      Print("CustomSymbolSetInteger() failed. Error ", GetLastError());

//--- 수정된 사용자 정의 심볼 속성을 저널에서 가져오고 인쇄합니다.
//--- (거래 모드, 스탑 주문 설치 수준 및 트레이딩 운영 중지 거리)
   ENUM_SYMBOL_TRADE_EXECUTION custom_exe_mode = (ENUM_SYMBOL_TRADE_EXECUTION)SymbolIn
   int custom_stops_level = (int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_ST
   int custom_freeze_level= (int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_TRADE_FR

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Deal execution mode: %s\n  Stops Level: %d\n  Freeze Level: %d",
               CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN,
               StringSubstr(EnumToString(custom_exe_mode), 23), custom_stops_level, cu

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Sleep(16);
      //--- Del을 누르면 생성된 사용자 정의 심볼이 삭제됩니다.
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
      }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Deal execution mode: INSTANT
     Stops Level: 0
     Freeze Level: 0
   Custom symbol 'EURUSD.C' based on 'EURUSD':
     Deal execution mode: MARKET
     Stops Level: 10
     Freeze Level: 3
   */
  }
//+------------------------------------------------------------------+
//| Create a custom symbol, return an error code                     |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| Remove a custom symbol                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- 성공
   return(true);
  }
```

추가 참조

SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetDouble

사용자 지정 심볼의 실제 유형 속성 값을 설정.

```
bool  CustomSymbolSetDouble(
   const string              symbol_name,      // 심볼 이름
   ENUM_SYMBOL_INFO_DOUBLE   property_id,      // 속성 ID
   double                    property_value    // 속성 값
   );
```

매개변수

symbol_name

[in] 사용자 지정 심볼 이름.

property_id

[in] 심볼 속성 ID. 값은 ENUM_SYMBOL_INFO_DOUBLE 열거 값 중 하나일 수 있습니다.

property_value

[in] 속성 값을 포함하는 double 유형 변수.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

심볼 사양에서 다음 속성 중 하나가 변경되면 사용자 지정 심볼의 분 및 틱 내역이 완전히 제거됩니다.

- SYMBOL_POINT – 1 포인트 값
- SYMBOL_TRADE_TICK_SIZE – 최소 허용 가격 변동을 지정하는 틱 값
- SYMBOL_TRADE_TICK_VALUE – 수익성 있는 포지션에 대한 1틱 가격 변동 값

사용자 지정 심볼 내역을 삭제한 후 터미널은 업데이트된 속성을 사용하여 새 내역을 생성하려고 시도합니다. 사용자 지정 심볼 속성을 수동으로 변경할 때도 마찬가지입니다.

예:

```
//+------------------------------------------------------------------+
//|                                        CustomSymbolSetDouble.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // 사용자 지정 심볼 명
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 심볼이 생성될 그룹의 이름
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // 사용자 정의 심볼의 기반이 되는 심볼 명

//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 사용자 정의 심볼의 기반이 되는 심볼의 속성을 저널에서 가져오고 출력합니다.
//--- (거래 실행을 위한 최소 거래량, 최대 거래량, 최소 거래량 변경 단계)
   double origin_vol_min = SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_MIN);
   double origin_vol_max = SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_MAX);
   double origin_vol_step= SymbolInfoDouble(CUSTOM_SYMBOL_ORIGIN, SYMBOL_VOLUME_STEP);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Volume Min: %.2f\n  Volume Max: %.2f\n  Volume Step: %.2f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               origin_vol_min, origin_vol_max, origin_vol_step);

//--- 사용자 정의 심볼 속성에 대한 다른 값을 설정합니다.
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MIN, 0.1);
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MAX, 1000);
   res &=CustomSymbolSetDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_STEP, 0.1);

//--- 속성을 설정할 때 오류가 발생하면 저널에 해당 메시지를 표시합니다.
   if(!res)
      Print("CustomSymbolSetDouble() failed. Error ", GetLastError());

//--- 수정된 사용자 정의 심볼 속성을 저널에서 가져오고 인쇄합니다.
//--- (거래 실행을 위한 최소 거래량, 최대 거래량, 최소 거래량 변경 단계)
   double custom_vol_min = SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MIN);
   double custom_vol_max = SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_MAX);
   double custom_vol_step= SymbolInfoDouble(CUSTOM_SYMBOL_NAME, SYMBOL_VOLUME_STEP);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Volume Min: %.2f\n  Volume Max: %.2f\n  Volume Step: %.2f",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               custom_vol_min, custom_vol_max, custom_vol_step);

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Sleep(16);
       //--- Del 을 누르면 생성된 사용자 정의 심볼이 삭제됩니다.
       if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
         {
          if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
             PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
          break;
         }
      }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Volume Min: 0.01
     Volume Max: 500.00
     Volume Step: 0.01
   Custom symbol 'EURUSD' based on 'EURUSD.C':
     Volume Min: 0.10
     Volume Max: 1000.00
     Volume Step: 0.10
   */
  }
//+------------------------------------------------------------------+
//| Create a custom symbol, return an error code                     |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| Remove a custom symbol                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- 성공
   return(true);
  }
```

추가 참조

SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetString

사용자 지정 심볼의 문자열 유형 속성 값을 설정.

```
bool  CustomSymbolSetString(
   const string               symbol_name,      // 심볼 이름
   ENUM_SYMBOL_INFO_STRING    property_id,      // 속성 ID
   string                     property_value    // 속성 값
   );
```

매개변수

symbol_name

[in] 사용자 지정 심볼 이름.

property_id

[in] 심볼 속성 ID. 값은 ENUM_SYMBOL_INFO_STRING 열거 값 중 하나일 수 있습니다.

property_value

[in] 속성 값을 포함하는 문자열 유형 변수.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

심볼 사양에서 SYMBOL_FORMULA 속성(사용자 지정 심볼 가격 구성에 대한 방정식 설정)을 변경하면 사용자 지정 심볼의 분 및 틱이 완전히 제거됩니다. 사용자 지정 심볼 내역을 삭제한 후 터미널은 새 방정식을 사용하여 새 내역을 생성하려고 시도합니다. 사용자 지정 심볼 방정식을 수동으로 변경할 때도 마찬가지 입니다.

예:

```
//+------------------------------------------------------------------+
//|                                        CustomSymbolSetString.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // 사용자 지정 심볼 명
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 심볼이 생성될 그룹의 이름
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // 사용자 정의 심볼의 기반이 되는 심볼 명

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 사용자 정의 심볼의 기반이 되는 심볼의 속성을 저널에서 가져오고 출력합니다.
//--- (거래 실행을 위한 최소 거래량, 최대 거래량, 최소 거래량 변경 단계)
   string origin_basis    = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_BASIS);
   string origin_category = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_CATEGORY);
   string origin_formula  = SymbolInfoString(CUSTOM_SYMBOL_ORIGIN, SYMBOL_FORMULA);

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Basis: %s\n  Category: %s\n  Formula: %s",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               origin_basis, origin_category, origin_formula);

//--- 사용자 정의 심볼 속성에 대한 다른 값을 설정합니다.
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_BASIS, CUSTOM_SYMBOL_ORIGIN);
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_CATEGORY, "FX");
   res &=CustomSymbolSetString(CUSTOM_SYMBOL_NAME, SYMBOL_FORMULA, ("1.0 / "+CUSTOM_SY

//--- 속성을 설정할 때 오류가 발생하면 저널에 해당 메시지를 표시합니다.
   if(!res)
      Print("CustomSymbolSetString() failed. Error ", GetLastError());

//--- 수정된 사용자 정의 심볼 속성을 저널에서 가져오고 인쇄합니다.
//--- (거래 실행을 위한 최소 거래량, 최대 거래량, 최소 거래량 변경 단계)
   string custom_basis    = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_BASIS);
   string custom_category = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_CATEGORY);
   string custom_formula  = SymbolInfoString(CUSTOM_SYMBOL_NAME, SYMBOL_FORMULA);

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Basis: %s\n  Category: %s\n  Formula: %s",
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               custom_basis, custom_category, custom_formula);

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del을 누르면 생성된 사용자 정의 심볼이 삭제됩니다.
```

© 2000-2025, MetaQuotes Ltd.

```
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Basis:
     Category:
     Formula:
   Custom symbol 'EURUSD' based on 'EURUSD.C':
     Basis: EURUSD
     Category: FX
     Formula: 1.0 / EURUSD
   */
  }
//+------------------------------------------------------------------+
//| 사용자 지정 심볼 생성, 에러 코드 반환                                |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| 사용자 정의 심볼 삭제                                             |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
       return(false);
      }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
      {
       PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastErro
       return(false);
      }
//--- 성공
   return(true);
  }
```

추가 참조

SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetMarginRate

사용자 지정 심볼의 순서 순서 유형 및 방향에 따라 마진율을 설정.

```
bool  CustomSymbolSetMarginRate(
   const string       symbol_name,              // 심볼 이름
   ENUM_ORDER_TYPE    order_type,               // 순서 유형
   double             initial_margin_rate,      // 초기 마진율
   double             maintenance_margin_rate   // 유지 마진율
   );
```

매개변수

symbol_name

[in] 사용자 지정 심볼 이름.

order_type

[in] 순서 유형.

initial_margin_rate

[in] 초기 마진율을 가진 double 유형 변수. 최초 마진은 적절한 방향의 1 lot 거래에 대한 보증금입니다. 최초 마진에 이율을 곱한 금액을 특정 유형의 주문 시 계좌에 적립해야 할 금액을 받습니다.

maintenance_margin_rate

[in] 유지 마진율을 가진 double 유형 변수. 유지 마진은 1 lot의 개방 위치를 적절한 방향으로 유지하기 위한 최소 금액입니다. 유지 마진에 이율을 곱한 금액을 특정 유형의 주문이 활성화된 후 당좌에 적립해야 할 금액을 받습니다.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

예:

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetMarginRate.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"  // 사용자 지정 심볼 명
#define   CUSTOM_SYMBOL_PATH     "Forex"        // 심볼이 생성될 그룹의 이름
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()       // 사용자 정의 심볼의 기반이 되는 심볼 명

#define   INITIAL_MARGIN_RATE       1.5         // 초기 마진 율
#define   MAINTENANCE_MARGIN_RATE   1.5         // 유지 마진 율
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 사용자 정의 심볼의 기반이 되는 심볼의 속성을 저널에서 가져오고 출력합니다.
//--- (매수 및 매도 주문에 대한 초기 및 유지 마진율)
   double initial_margin_rate_buy=0;
   double maintenance_margin_rate_buy=0;
   double initial_margin_rate_sell=0;
   double maintenance_margin_rate_sell=0;

   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_ORIGIN, ORDER_TYPE_BUY, initial_margin_rate_b
      return;
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_ORIGIN, ORDER_TYPE_SELL,initial_margin_rate_s
      return;

   PrintFormat("The '%s' symbol from which the custom '%s' was created:\n"+
               "  Buy order initial margin rate: %f\n  Buy order maintenance margin ra
               "  Sell order initial margin rate: %f\n  Sell order maintenance margin
               CUSTOM_SYMBOL_ORIGIN, CUSTOM_SYMBOL_NAME,
               initial_margin_rate_buy, maintenance_margin_rate_buy, initial_margin_ra

//--- 사용자 정의 심볼 속성에 대한 다른 값을 설정합니다.
   ResetLastError();
   bool res=true;
   res &=CustomSymbolSetMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_BUY, INITIAL_MARGIN_
   res &=CustomSymbolSetMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_SELL,INITIAL_MARGIN_

//--- 속성을 설정할 때 오류가 발생하면 저널에 해당 메시지를 표시합니다.
   if(!res)
      Print("CustomSymbolSetMarginRate() failed. Error ", GetLastError());

//--- 수정된 사용자 정의 심볼 속성을 저널에서 가져오고 인쇄합니다.
//--- (매수 및 매도 주문에 대한 초기 및 유지 마진율)
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_BUY, initial_margin_rate_buy
      return;
   if(!GetSymbolMarginRate(CUSTOM_SYMBOL_NAME, ORDER_TYPE_SELL,initial_margin_rate_sel
      return;

   PrintFormat("Custom symbol '%s' based on '%s':\n"+
               "  Buy order initial margin rate: %f\n  Buy order maintenance margin ra
```

© 2000-2025, MetaQuotes Ltd.

```
                "  Sell order initial margin rate: %f\n  Sell order maintenance margin
                CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_ORIGIN,
                initial_margin_rate_buy, maintenance_margin_rate_buy, initial_margin_ra

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex

//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del을 누르면 생성된 사용자 정의 심볼이 삭제됩니다.
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   The 'EURUSD' symbol from which the custom 'EURUSD.C' was created:
     Buy order initial margin rate: 1.000000
     Buy order maintenance margin rate: 0.000000
     Sell order initial margin rate: 1.000000
     Sell order maintenance margin rate: 0.000000
   Custom symbol 'EURUSD.C' based on 'EURUSD':
     Buy order initial margin rate: 1.500000
     Buy order maintenance margin rate: 1.500000
     Sell order initial margin rate: 1.500000
     Sell order maintenance margin rate: 1.500000
   */
  }
//+------------------------------------------------------------------+
//| Create a custom symbol, return an error code                     |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| Remove a custom symbol                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastErro
      return(false);
     }
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| Return margin ratios                                             |
//+------------------------------------------------------------------+
bool GetSymbolMarginRate(const string symbol, const ENUM_ORDER_TYPE order_type, double
  {
   ResetLastError();
   if(!SymbolInfoMarginRate(symbol, order_type, initial_margin_rate, maintenance_margi
     {
      PrintFormat("%s: SymbolInfoMarginRate(%s, %s) failed. Error %d",__FUNCTION__, sy
      return false;
     }
   return true;
  }
```

추가 참조

SymbolInfoMarginRate

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetSessionQuote

지정된 심볼 및 평일에 대해 지정된 시세 세션의 시작 및 종료 시간을 설정합니다.

```
bool  CustomSymbolSetSessionQuote(
   const string      symbol_name,          // 심볼 이름
   ENUM_DAY_OF_WEEK  day_of_week,          // 평일
   uint              session_index,        // 세션 인덱스
   datetime          from,                 // 세션 시작 시간
   datetime          to                    // 세션 종료 시간
   );
```

매개변수

symbol_name

[in] 사용자 지정 심볼 이름.

ENUM_DAY_OF_WEEK

[in] 평일, ENUM_DAY_OF_WEEK 열거 값.

uint

[in] 시작 시간과 종료 시간을 설정할 세션의 인덱스. 세션 인덱싱은 0부터 시작합니다.

from

[in] 00:00부터의 세션 시작 시간(초), 변수의 데이터 값은 무시됩니다.

to

[in] 00:00부터 세션 종료 시간(초), 변수의 데이터 값은 무시됩니다.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

지정된 *session_index*를 가진 세션이 이미 있는 경우, 이 함수는 단순히 세션의 시작과 끝을 편집합니다.

세션에 대해 0 시작 및 종료 매개변수가 전달될 경우(*from=0* and *to=0*), *session_index*가 아래로 이동하는 동안 해당 세션은 삭제됩니다.

세션은 순차적으로 추가할 수 있습니다. 즉 인덱가 0인 세션이 이미 있는 경우에만 *session_index=1*을 추가할 수 있습니다. 이 규칙이 위반되면 새 세션이 생성되지 않고 함수 자체는 'false'를 반환합니다.

예:

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetSessionQuote.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
```

© 2000-2025, MetaQuotes Ltd.

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"           // 사용자 지정 심볼 명
#define   CUSTOM_SYMBOL_PATH     "Forex"                 // 심볼이 생성될 그룹의 이름
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()                // 사용자 정의 심볼의 기반이 되는

#define   SESSION_0_FROM         D'1970.01.01 00:15:00'  // 세션 0 시작 시간
#define   SESSION_0_TO           D'1970.01.01 11:59:00'  // 세션 0 종료 시간
#define   SESSION_1_FROM         D'1970.01.01 12:15:00'  // 세션 1 시작 시간
#define   SESSION_1_TO           D'1970.01.01 23:59:00'  // 세션 1 종료 시간

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 기본 심볼과 세션 인덱스가 포함된 헤더를 인쇄하고
//--- 월요일부터 금요일까지 요일별 루프에서 각 쿼트 세션의 시작 및 종료 시간을 저널에 출력합니다.
   for(int session=0; session<2; session++)
     {
      PrintFormat("Quote session %d of '%s' symbol from which the custom '%s' was crea
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_ORIGIN, (ENUM_DAY_OF_WEEK)day_of_we
     }

//--- 두 세션의 루프에서
   bool res=true;
   for(int session=0; session<2; session++)
     {
      datetime from = SESSION_0_FROM;
      datetime to   = SESSION_0_TO;
      if(session>0)
        {
         from = SESSION_1_FROM;
         to   = SESSION_1_TO;
        }
      //--- 각 요일의 맞춤 심볼에 대한 쿼트 세션 시간 설정
      ResetLastError();
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         res &=CustomSymbolSetSessionQuote(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_o
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 세션 설정 시 오류가 발생한 경우 저널에 해당 메시지를 표시합니다.
   if(!res)
      Print("CustomSymbolSetSessionQuote() failed. Error ", GetLastError());

//--- 사용자 정의 심볼과 세션 인덱스로 헤더를 인쇄하고
//--- 월요일부터 금요일까지 요일별 루프에서 각 쿼트 세션의 시작 및 종료 시간을 저널에 출력합니다.
   for(int session=0; session<2; session++)
     {
      PrintFormat("Quote session %d of custom symbol '%s' based on '%s'", session, CUS
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_of_week
     }

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del을 누르면 생성된 사용자 정의 심볼이 삭제됩니다.
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   Quote session 0 of 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   Quote session 1 of 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     Session not set
   - Tuesday    Session not set
   - Wednesday  Session not set
   - Thursday   Session not set
   - Friday     Session not set
   Quote session 0 of custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     00:15 - 11:59
   - Tuesday    00:15 - 11:59
   - Wednesday  00:15 - 11:59
   - Thursday   00:15 - 11:59
```

© 2000-2025, MetaQuotes Ltd.

```
   - Friday     00:15 - 11:59
   Quote session 1 of custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     12:15 - 23:59
   - Tuesday    12:15 - 23:59
   - Wednesday  12:15 - 23:59
   - Thursday   12:15 - 23:59
   - Friday     12:15 - 23:59
   */
  }
//+------------------------------------------------------------------+
//| Create a custom symbol, return an error code                     |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| 사용자 정의 심볼 삭제                                                  |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 지정된 쿼트 세션의 시작 및 종료 시간 보내기                          |
//| 지정된 심볼 및 요일에 대한 저널                                     |
//+------------------------------------------------------------------+
void SymbolInfoSessionQuotePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- 쿼트 세션의 시작과 끝을 기록하는 변수를 선언합니다.
   datetime date_from;  // session start time
   datetime date_to;    // session end time

//--- 열거형 상수로부터 요일 이름을 생성합니다.
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- 심볼 및 요일별로 쿼트 세션에서 데이터를 가져옵니다.
   if(!SymbolInfoSessionQuote(symbol, day_of_week, session_index, date_from, date_to))
     {
      int err=GetLastError();
      string message=(err==4307 ? StringFormat("- %-10s Session not set", week_day) :
                      StringFormat("SymbolInfoSessionQuote(%s, %s, session %d) failed
      Print(message);
      return;
     }

//--- 지정된 쿼트 세션에 대한 데이터를 저널로 보냅니다.
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

추가 참조

SymbolInfoSessionQuote, Symbol info, TimeToStruct, Date structure

© 2000-2025, MetaQuotes Ltd.

CustomSymbolSetSessionTrade

지정된 심볼 및 평일에 대해 지정된 트레이딩 세션의 시작 및 종료 시간을 설정합니다.

```
bool  CustomSymbolSetSessionTrade(
   const string      symbol_name,          // 심볼 이름
   ENUM_DAY_OF_WEEK  day_of_week,          // 평일
   uint              session_index,        // 세션 인덱스
   datetime          from,                 // 세션 시작 시간
   datetime          to                    // 세션 종료 시간
   );
```

매개변수

symbol_name

[in] 사용자 지정 심볼 이름.

ENUM_DAY_OF_WEEK

[in] 평일, ENUM_DAY_OF_WEEK 열거 값.

uint

[in] 시작 시간과 종료 시간을 설정할 세션의 인덱스. 세션 인덱싱은 0부터 시작합니다.

from

[in] 00:00부터의 세션 시작 시간(초), 변수의 데이터 값은 무시됩니다.

to

[in] 00:00부터 세션 종료 시간(초), 변수의 데이터 값은 무시됩니다.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

지정된 *session_index*를 가진 세션이 이미 있는 경우, 이 함수는 단순히 세션의 시작과 끝을 편집합니다.

세션에 대해 0 시작 및 종료 매개변수가 전달될 경우(*from=0* and *to=0*), *session_index*가 아래로 이동하는 동안 해당 세션은 삭제됩니다.

세션은 순차적으로 추가할 수 있습니다. 즉 인덱가 0인 세션이 이미 있는 경우에만 *session_index=1*을 추가할 수 있습니다. 이 규칙이 위반되면 새 세션이 생성되지 않고 함수 자체는 'false'를 반환합니다.

예:

```
//+------------------------------------------------------------------+
//|                                  CustomSymbolSetSessionTrade.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
```

© 2000-2025, MetaQuotes Ltd.

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"           // 사용자 지정 심볼 명
#define   CUSTOM_SYMBOL_PATH     "Forex"                 // 심볼이 생성될 그룹의 이름
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()                // 사용자 정의 심볼의 기반이 되는

#define   SESSION_0_FROM         D'1970.01.01 00:15:00'  // 세션 0 시작 시간
#define   SESSION_0_TO           D'1970.01.01 11:59:00'  // 세션 0 종료 시간
#define   SESSION_1_FROM         D'1970.01.01 12:15:00'  // 세션 1 시작 시간
#define   SESSION_1_TO           D'1970.01.01 23:59:00'  // 세션 1 종료 시간

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 기본 심볼과 세션 인덱스가 포함된 헤더를 인쇄하고
//--- 월요일부터 금요일까지 요일별 루프에서 각 거래 세션의 시작 및 종료 시간을 저널에 출력합니다.
   for(int session=0; session<2; session++)
     {
      PrintFormat("Trade session %d of the '%s' symbol from which the custom '%s' was
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_ORIGIN, (ENUM_DAY_OF_WEEK)day_of_we
     }

//--- 두 세션의 루프에서
   bool res=true;
   for(int session=0; session<2; session++)
     {
      datetime from = SESSION_0_FROM;
      datetime to   = SESSION_0_TO;
      if(session>0)
        {
         from = SESSION_1_FROM;
         to   = SESSION_1_TO;
        }
      //--- 각 요일의 맞춤 심볼에 대한 쿼트 세션 시간 설정
      ResetLastError();
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         res &=CustomSymbolSetSessionQuote(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_o
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 세션 설정 시 오류가 발생한 경우 저널에 해당 메시지를 표시합니다.
   if(!res)
      Print("CustomSymbolSetSessionTrade() failed. Error ", GetLastError());

//--- 사용자 정의 심볼과 세션 인덱스로 헤더를 인쇄하고
//--- 월요일부터 금요일까지 요일별 루프에서 각 거래 세션의 시작 및 종료 시간을 저널에 출력합니다.
   for(int session=0; session<2; session++)
     {
      PrintFormat("Trade session %d of a custom symbol '%s' based on '%s'", session, C
      for(int day_of_week=MONDAY; day_of_week<SATURDAY; day_of_week++)
         SymbolInfoSessionQuotePrint(CUSTOM_SYMBOL_NAME, (ENUM_DAY_OF_WEEK)day_of_week
     }

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- Del을 누르면 생성된 사용자 정의 심볼이 삭제됩니다.
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME);
         break;
        }
     }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   Trade session 0 of the 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     00:15 - 23:55
   - Tuesday    00:15 - 23:55
   - Wednesday  00:15 - 23:55
   - Thursday   00:15 - 23:55
   - Friday     00:15 - 23:55
   Trade session 1 of the 'EURUSD' symbol from which the custom 'EURUSD.C' was created
   - Monday     Session not set
   - Tuesday    Session not set
   - Wednesday  Session not set
   - Thursday   Session not set
   - Friday     Session not set
   Trade session 0 of a custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     00:15 - 11:59
   - Tuesday    00:15 - 11:59
   - Wednesday  00:15 - 11:59
   - Thursday   00:15 - 11:59
```

© 2000-2025, MetaQuotes Ltd.

```
   - Friday     00:15 - 11:59
   Trade session 1 of a custom symbol 'EURUSD.C' based on 'EURUSD'
   - Monday     12:15 - 23:59
   - Tuesday    12:15 - 23:59
   - Wednesday  12:15 - 23:59
   - Thursday   12:15 - 23:59
   - Friday     12:15 - 23:59
   */
  }
//+------------------------------------------------------------------+
//| 사용자 지정 심볼 생성, 에러 코드 반환                                |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| 사용자 정의 심볼 삭제                                             |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 지정된 쿼트 세션의 시작 및 종료 시간 보내기                          |
//| 지정된 심볼 및 요일에 대한 저널                                     |
//+------------------------------------------------------------------+
void SymbolInfoSessionTradePrint(const string symbol, const ENUM_DAY_OF_WEEK day_of_we
  {
//--- 쿼트 세션의 시작과 끝을 기록하는 변수를 선언합니다.
   datetime date_from;  // session start time
   datetime date_to;    // session end time

//--- 열거형 상수로부터 요일 이름을 생성합니다.
   string week_day=EnumToString(day_of_week);
   if(week_day.Lower())
      week_day.SetChar(0, ushort(week_day.GetChar(0)-32));

//--- 심볼 및 요일별로 쿼트 세션에서 데이터를 가져옵니다.
   if(!SymbolInfoSessionTrade(symbol, day_of_week, session_index, date_from, date_to))
     {
      int err=GetLastError();
      string message=(err==4307 ? StringFormat("- %-10s Session not set", week_day) :
                      StringFormat("SymbolInfoSessionTrade(%s, %s, session %d) failed
      Print(message);
      return;
     }

//--- 지정된 거래 세션에 대한 데이터를 저널로 보냅니다.
   PrintFormat("- %-10s %s - %s", week_day, TimeToString(date_from, TIME_MINUTES), Tir
  }
```

추가 참조

SymbolInfoSessionTrade, Symbol info, TimeToStruct, Date structure

© 2000-2025, MetaQuotes Ltd.

CustomRatesDelete

지정된 시간 간격 동안 사용자 지정 심볼의 가격 내역에서 모든 바를 삭제.

```
int  CustomRatesDelete(
   const string      symbol,        // 심볼 이름
   datetime          from,          // 시작일
   datetime          to             // 종료일
   );
```

매개변수

심볼

[in] 사용자 지정 심볼 이름.

from

[in] 가격 기록에서 제거할 특정 범위 내 첫 번째 바의 시간.

to

[in] 지정된 범위 내에서 가격 기록에서 제거할 마지막 바의 시간.

값 반환

오류가 발생한 경우 삭제된 바의 수 또는 -1.

예:

```
//+------------------------------------------------------------------+
//|                                            CustomRatesDelete.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // 사용자 지정 심볼명
#define   CUSTOM_SYMBOL_PATH     "Forex"           // 심볼을 생성할 그룹 명
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // 사용자 정의 심볼의 기반이 되는 심볼

#define   DATARATES_COUNT        4                 // 저멀에 보내진 바의 개수

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 표준 심볼 바의 개수를 가져옵니다.
   int bars=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);

//--- 표준 심볼 분 시간대의 모든 바의 데이터를 MqlRates 배열로 가져옵니다.
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- 복사된 데이터를 사용자 정의 심볼의 분 히스토리로 설정합니다.
   ResetLastError();
   if(CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates)<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- 과거 데이터를 업데이트한 후 사용자 정의 심볼의 바의 개수를 가져옵니다.
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- 사용자 정의 심볼 분 시간대의 모든 바의 데이터를 MqlRates 배열로 가져옵니다.
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- 저널에 있는 사용자 정의 심볼 분 히스토리의 마지막 DATARATES_COUNT 바를 출력합니다.
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- 사용자 정의 심볼 분 히스토리에서 어미에서 두 번째 데이터 바를 삭제합니다.
   datetime time_from= rates[bars-3].time;
   datetime time_to  = rates[bars-2].time;
   ResetLastError();
   int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, time_from, time_to);
   if(deleted<0)
     {
      PrintFormat("CustomRatesDelete(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- 과거 데이터 바 두 개를 삭제한 후 사용자 정의 심볼 바의 개수를 다시 가져옵니다.
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- 사용자 정의 심볼 분 차트 주기의 나머지 모든 바 데이터를 다시 가져옵니다.
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- 저널에 업데이트된 사용자 정의 바 분 기록의 마지막 DATARATES_COUNT 바를 인쇄합니다.
   PrintFormat("\nLast %d bars after applying CustomRatesDelete() with %d deleted bars
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
//--- Del 키를 누르면 생성된 사용자 정의 심볼 및 해당 데이터가 삭제됩니다.
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
//--- 바 데이터 삭제
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d

         //--- 틱 데이터 삭제
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- 심볼 삭제
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   Last 4 bars of the custom symbol's minute history:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 20:53:00 1.07341 1.07347 1.07336 1.07343            38        0
```

© 2000-2025, MetaQuotes Ltd.

```
   [1] 2024.06.18 20:54:00 1.07344 1.07354 1.07344 1.07353            21         0
   [2] 2024.06.18 20:55:00 1.07353 1.07362 1.07351 1.07356            32         0
   [3] 2024.06.18 20:56:00 1.07356 1.07358 1.07352 1.07354            24         0

   Last 4 bars after applying CustomRatesDelete() with 2 deleted bars:
                    [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 20:51:00 1.07357 1.07358 1.07347 1.07349            25         0
   [1] 2024.06.18 20:52:00 1.07349 1.07350 1.07336 1.07341            31         0
   [2] 2024.06.18 20:53:00 1.07341 1.07347 1.07336 1.07343            38         0
   [3] 2024.06.18 20:56:00 1.07356 1.07358 1.07352 1.07354            24         0
   */
  }
//+------------------------------------------------------------------+
//| Create a custom symbol, return an error code                     |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| Remove a custom symbol                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- 성공
   return(true);
  }
```

추가 참조

CustomRatesReplace, CustomRatesUpdate, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomRatesReplace

지정된 시간 간격 내에 사용자 지정 심볼의 가격 내역을 MqlRates 유형 배열의 데이터로 완전히 대체합니다.

```
int  CustomRatesReplace(
   const string     symbol,              // 심볼 이름
   datetime         from,                // 시작일
   datetime         to,                  // 종료일
   const MqlRates&  rates[],             // 사용자 지정 심볼에 적용할 데이터의 배열
   uint             count=WHOLE_ARRAY    // 사용할 rates[] 배열 요소의 수
   );
```

매개변수

심볼

[in] 사용자 지정 심볼 이름.

from

[in] 지정된 범위 내에서 가격 내역에 있는 첫 번째 막대의 업데이트 시간.

to

[in] 지정된 범위 내에서 가격 내역에 있는 마지막 막대의 업데이트 시간.

rates[]

[in] M1에 대한 MqlRates 유형 내역 데이터의 배열.

count=WHOLE_ARRAY

[in] 교체를 위해 사용할 *rates[]* 배열 요소의 수. WHOLE_ARRAY 모든 *rates[]* 배열 요소를 교체용으로 사용해야 함을 의미합니다.

값 반환

오류가 발생한 경우 업데이트된 수 또는 -1.

주의

rates[] 배열의 막대가 지정된 범위를 벗어나면 무시됩니다. 그러한 막대가 이미 가격 내역에 존재하고 주어진 범위에 들어간 경우, 대체됩니다. 지정된 범위를 벗어난 현재 가격 내역의 다른 모든 막대는 변경되지 않습니다. *rates[]* 배열 데이터는 OHLC 가격에 대해 정확해야 하며 막대 오프닝 시간은 M1 시간 프레임에 해당되어야 합니다.

예:

```
//+------------------------------------------------------------------+
//|                                          CustomRatesReplace.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // 사용자 지정 심볼명
#define   CUSTOM_SYMBOL_PATH     "Forex"           // 심볼을 생성할 그룹 명
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // 사용자 정의 심볼의 기반이 되는 심볼

#define   DATARATES_COUNT        4                 // 저널에 보내진 바의 개수

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 표준 심볼 바의 개수를 가져옵니다.
   int bars=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);

//--- 표준 심볼 분 시간대의 모든 바의 데이터를 MqlRates 배열로 가져옵니다.
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- 복사된 데이터를 사용자 정의 심볼의 분 히스토리로 설정합니다.
   ResetLastError();
   if(CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates)<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- 과거 데이터를 업데이트한 후 사용자 정의 심볼의 바의 개수를 가져옵니다.
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- 사용자 정의 심볼 분 시간대의 모든 바의 데이터를 MqlRates 배열로 가져옵니다.
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- 저널에 있는 사용자 정의 심볼 분 히스토리의 마지막 DATARATES_COUNT 바를 출력합니다.
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- 사용자 정의 기호 분 히스토리에서 어미에서 두 번째 데이터 바를 변경합니다.
   datetime time_from= rates[bars-3].time;
   datetime time_to  = rates[bars-2].time;

//--- 끝에서 두 번째 바의 모든 가격을 'rates' 배열에 있는 이 바의 시가와 동일하게 만듭니다.
   rates[bars-3].high=rates[bars-3].open;
   rates[bars-3].low=rates[bars-3].open;
   rates[bars-3].close=rates[bars-3].open;

   rates[bars-2].high=rates[bars-2].open;
   rates[bars-2].low=rates[bars-2].open;
   rates[bars-2].close=rates[bars-2].open;

//--- 기존 바를 수정된 'rates' 배열의 데이터로 바꿉니다.
   ResetLastError();
   int replaced=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(replaced<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- 과거 데이터의 두 바를 변경한 후 사용자 정의 심볼 바의 개수를 다시 가져옵니다.
   bars=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);

//--- 사용자 정의 심볼 분 찰 주기의 모든 바의 데이터를 다시 가져옵니다.
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars, rates)!=bars)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- 저널에 업데이트된 사용자 정의 바 분 히스토리의 마지막 DATARATES_COUNT 바를 인쇄합니다.
   PrintFormat("\nLast %d bars after applying CustomRatesUpdate() with %d replaced bar
   ArrayPrint(rates, digits, NULL, bars-DATARATES_COUNT, DATARATES_COUNT);

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Sleep(16);
//--- Del 키를 누르면 생성된 사용자 정의 심볼 및 해당 데이터가 삭제됩니다.
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
//--- 바 데이터 삭제
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d

         //--- 틱 데이터 삭제
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- 심볼 삭제
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   사용자 정의 심볼 분 히스토리의 마지막 4개 바:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.07.29 13:37:00 1.08394 1.08396 1.08388 1.08390            16        1
   [1] 2024.07.29 13:38:00 1.08389 1.08400 1.08389 1.08398            35        1
   [2] 2024.07.29 13:39:00 1.08398 1.08410 1.08394 1.08410            29        1
   [3] 2024.07.29 13:40:00 1.08409 1.08414 1.08408 1.08414            14        1

  250820개의 바에 대체된 CustomRatesUpdate()를 적용한 후 마지막 4개 바:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.07.29 13:37:00 1.08394 1.08396 1.08388 1.08390            16        1
   [1] 2024.07.29 13:38:00 1.08389 1.08389 1.08389 1.08389            35        1
   [2] 2024.07.29 13:39:00 1.08398 1.08398 1.08398 1.08398            29        1
   [3] 2024.07.29 13:40:00 1.08409 1.08414 1.08408 1.08414            14        1
   */
  }
//+------------------------------------------------------------------+
//| Create a custom symbol, return an error code                     |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| Remove a custom symbol                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastErro
      return(false);
     }
//--- 성공
   return(true);
  }
```

추가 참조

CustomRatesDelete, CustomRatesUpdate, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomRatesUpdate

누락된 막대를 사용자 지정 심볼 내역에 추가하고 기존 데이터를 MqlRates 유형 배열의 데이터로 바꿉니다.

```
int  CustomRatesUpdate(
   const string     symbol,             // 사용자 지정 심볼 이름
   const MqlRates&  rates[],            // 사용자 지정 심볼에 적용할 데이터의 배열
   uint             count=WHOLE_ARRAY   // 사용할 rates[] 배열 요소의 수
   );
```

매개변수

심볼

[in] 사용자 지정 심볼 이름.

rates[]

[in] M1에 대한 MqlRates 유형 내역 데이터의 배열입니다.

count=WHOLE_ARRAY

[in] 업데이트에 사용할 *rates[]* 배열 요소의 수. WHOLE_ARRAY는 모든 *rates[]* 배열 요소를 사용해야 함을 의미합니다.

값 반환

오류가 발생한 경우 업데이트된 수 또는 -1.

주의

현재 사용자 지정 심볼 내역에 *rates[]* 배열에 막대가 없으면 추가됩니다. 이러한 막대가 이미 있는 경우 대체됩니다. 현재 가격 내역의 다른 모든 막대는 변경되지 않습니다. *rates[]* 배열 데이터는 OHLC 가격에 대해 정확해야 하며 막대 오프닝 시간은 M1 시간 프레임에 해당되어야 합니다.

Example:

```
//+------------------------------------------------------------------+
//|                                              CustomRatesUpdate.mq5 |
//|                                 Copyright 2024, MetaQuotes Ltd.  |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"      // custom symbol name
#define   CUSTOM_SYMBOL_PATH     "Forex"            // name of the group, in which a sy
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()           // name of a symbol a custom one is

#define   DATARATES_COUNT        4                  // number of bars sent to the journ

//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- get the error code when creating a custom symbol
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- if the error code is not 0 (successful symbol creation) and not 5304 (symbol has
   if(create!=0 && create!=5304)
      return;

//--- get and print in the journal the number of standard symbol bars
   int bars_origin=Bars(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1);
   PrintFormat("The symbol '%s' from which the custom '%s' was created has %d bars of

//--- get and print in the journal the number of custom symbol bars
   int bars_custom=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);
   PrintFormat("Custom symbol '%s' created from symbol '%s' has %d bars of minute hist

//--- get the data of all bars of the standard symbol minute timeframe into the MqlRat
   MqlRates rates[]={};
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_ORIGIN, PERIOD_M1, 0, bars_origin, rates)!=bars_origin)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_OR
      return;
     }

//--- set the copied data to the minute history of the custom symbol
   ResetLastError();
   int updated=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(updated<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- get and print in the journal the number of custom symbol bars after adding histo
   bars_custom=Bars(CUSTOM_SYMBOL_NAME, PERIOD_M1);
   PrintFormat("\nAfter CustomRatesUpdate(), the custom symbol '%s' has %d bars of mi

//--- get the data of all bars of the custom symbol minute timeframe into the MqlRates
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars_custom, rates)!=bars_custom)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- print the last four bars of the custom symbol minute history in the journal
   int digits=(int)SymbolInfoInteger(CUSTOM_SYMBOL_NAME, SYMBOL_DIGITS);
   PrintFormat("Last %d bars of the custom symbol's minute history:", DATARATES_COUNT)
   ArrayPrint(rates, digits, NULL, bars_custom-DATARATES_COUNT, DATARATES_COUNT);

//--- replace the data in the MqlRates array with the one calculated using the equatio
   for(int i=0; i<bars_custom; i++)
     {
      rates[i].open  =(rates[i].open !=0  ? 1.0 / rates[i].open  : rates[i].open);
      rates[i].high  =(rates[i].high !=0  ? 1.0 / rates[i].high  : rates[i].high);
      rates[i].low   =(rates[i].low  !=0  ? 1.0 / rates[i].low   : rates[i].low);
      rates[i].close =(rates[i].close!=0  ? 1.0 / rates[i].close : rates[i].close);
     }

//--- set the modified data to the minute history of the custom symbol
   ResetLastError();
   updated=CustomRatesUpdate(CUSTOM_SYMBOL_NAME, rates);
   if(updated<0)
     {
      PrintFormat("CustomRatesUpdate(%s) failed. Error %d", CUSTOM_SYMBOL_NAME, GetLas
      return;
     }

//--- get the data of all bars of the custom symbol minute timeframe into the MqlRates
   ResetLastError();
   if(CopyRates(CUSTOM_SYMBOL_NAME, PERIOD_M1, 0, bars_custom, rates)!=bars_custom)
     {
      PrintFormat("CopyRates(%s, PERIOD_M1, 0, %d) failed. Error %d", CUSTOM_SYMBOL_NA
      return;
     }

//--- print the last four bars of the updated custom symbol minute history in the jou
   Print("\nLast %d bars after changing the custom symbol calculation formula:", DATAR
   ArrayPrint(rates, digits, NULL, bars_custom-DATARATES_COUNT, DATARATES_COUNT);

//--- display a hint about the script termination keys on the chart comment
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- wait for pressing the Esc or Del keys to exit in an endless loop
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
      //--- when pressing Del, delete the created custom symbol and its data
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
         //--- delete bar data
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- delete tick data
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- delete symbol
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- clear the chart before exiting
   Comment("");
   /*
   result:
   The symbol 'EURUSD' from which the custom 'EURUSD.C' was created has 250488 bars o
   Custom symbol 'EURUSD.C' created from symbol 'EURUSD' has 0 bars of minute history

   After CustomRatesUpdate(), the custom symbol 'EURUSD.C' has 250488 bars of minute h
   Last 4 bars of the custom symbol's minute history:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 11:14:00 1.07235 1.07239 1.07232 1.07239            24        0
   [1] 2024.06.18 11:15:00 1.07238 1.07239 1.07232 1.07235            44        0
   [2] 2024.06.18 11:16:00 1.07234 1.07238 1.07227 1.07234            37        0
   [3] 2024.06.18 11:17:00 1.07234 1.07234 1.07217 1.07225            41        0

   Last 4 bars after changing the custom symbol calculation formula:
                     [time]  [open]  [high]   [low] [close] [tick_volume] [spread] [rea
   [0] 2024.06.18 11:14:00 0.93253 0.93250 0.93256 0.93250            24        0
   [1] 2024.06.18 11:15:00 0.93251 0.93250 0.93256 0.93253            44        0
   [2] 2024.06.18 11:16:00 0.93254 0.93251 0.93260 0.93254            37        0
   [3] 2024.06.18 11:17:00 0.93254 0.93254 0.93269 0.93262            41        0
   */
  }
//+------------------------------------------------------------------+
//| Create a custom symbol, return an error code                     |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- define the name of a symbol a custom one is to be based on
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- if failed to create a custom symbol and this is not error 5304, report this in t
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
```

© 2000-2025, MetaQuotes Ltd.

```
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
      }
//--- successful
   return(error);
  }
//+------------------------------------------------------------------+
//| Remove a custom symbol                                           |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- hide the symbol from the Market Watch window
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- if failed to delete a custom symbol, report this in the journal and return 'fals
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- successful
   return(true);
  }
```

추가 참조

CustomRatesReplace, CustomRatesDelete, CopyRates

© 2000-2025, MetaQuotes Ltd.

CustomTicksAdd

MqlTick 유형의 배열에 있는 데이터를 사용자 지정 심볼의 가격 내역에 추가. 종합 시세 창에서 사용자 지정 심볼을 선택해야 합니다.

```
int  CustomTicksAdd(
   const string     symbol,              // 심볼 이름
   const MqlTick&   ticks[],             // 사용자 지정 심볼에 적용해야 하는 틱 데이터가 있는
   uint             count=WHOLE_ARRAY    // 사용할 ticks[] 배열 요소의 수
   );
```

매개변수

심볼

[in] 사용자 지정 심볼의 이름.

ticks[]

[in] MqlTick 유형의 틱 데이터 배열은 이전 데이터에서 최근 데이터까지 시간 순서대로 배열됩니다, 예를 들면 ticks[k].time_msc <= ticks[n].time_msc, if k<n.

count=WHOLE_ARRAY

[in] 추가하는데 사용할 ***ticks[]*** 배열 요소의 수. WHOLE_ARRAY 는 모든 ***ticks[]*** 배열 요소를 사용해야 함을 의미합니다.

값 반환

오류가 발생한 경우 추가된 틱 수 또는 -1.

추가 참고

CustomTicksAdd 기능은 종합 시세 창에 열린 사용자 지정 심볼에 대해서만 작동합니다. 종합 시세(Market Watch)에서 심볼이 선택되지 않은 경우, CustomTicksReplace을 사용하여 틱을 추가해야 합니다.

CustomTicksAdd 기능을 통해 브로커의 서버에서 제공된 것처럼 틱을 전송할 수 있습니다. 데이터는 체크 표시 데이터베이스에 직접 기록되지 않고 종합 시세(Market Watch) 창으로 전송됩니다. 그런 다음 종합 시세의 틱을 데이터베이스에 저장합니다. 한 함수 호출 중에 전송되는 데이터 양이 많으면 리소스 사용을 줄이기 위해 함수 동작이 변경됩니다. 256개 이상의 틱이 통과되면 데이터가 두 부분으로 나뉩니다. 첫 번째, 즉 큰 부분은 CustomTicksReplace에서 수행한 것 처럼 틱 데이터베이스에 직접 기록됩니다. 128개의 틱이 포함된 두 번째 부분은 종합 시세 창으로 전달되며, 이 창에서 터미널은 틱을 데이터베이스에 저장합니다.

MqlTick 구조에는 시간 값(초 단위의 눈금 시간)과 1970년 1월 1일부터 카운트되는 time_msc (밀리초 단위의 눈금 시간)의 두 필드가 있습니다. 추가된 틱의 이러한 필드는 다음 순서로 처리됩니다.

1. ticks[k].time_msc!=0인 경우, ticks[k].time 필드를 채우는데 사용하며, 예를 들어 틱에 대해 ticks[k].time=ticks[k].time_msc/1000 (정수 부분)이 설정
2. ticks[k].time_msc==0 이고 ticks[k].time!=0인 경우, 밀리초 단위의 시간은 1000을 곱하여 얻습니다, 예를 들어 ticks[k].time_msc=ticks[k].time*1000
3. ticks[k].time_msc==0 이고 ticks[k].time==0인 경우, 현재 거래 서버 시간은 CustomTicksAdd 호출 시점 현재 밀리초까지 이 필드에 기록됩니다.

ticks[k].bid, ticks[k].ask, ticks[k].last 또는 **ticks[k].**volume의 값이 0보다 크면 적절한 플래그 조합이 ticks[k].flags 필드에 기록됩니다:

© 2000-2025, MetaQuotes Ltd.

- TICK_FLAG_BID - 틱이 입찰 가격을 변경했습니다
- TICK_FLAG_ASK - 틱이 요청 가격을 변경했습니다
- TICK_FLAG_LAST - 틱이 마지막 거래 가격을 변경했습니다
- TICK_FLAG_VOLUME - 틱이 볼륨을 변경했습니다

필드 값이 0보다 작거나 같으면 해당 플래그가 ticks[k].flags 필드에 기록되지 않습니다.

TICK_FLAG_BUY 및 TICK_FLAG_SELL 플래그는 사용자 지정 심볼의 기록에 추가되지 않습니다.

예:

```
//+------------------------------------------------------------------+
//|                                                CustomTicksAdd.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // 사용자 지정 심볼명
#define   CUSTOM_SYMBOL_PATH     "Forex"           // 심볼을 생성할 그룹 명
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // 사용자 정의 심볼의 기반이 되는 심볼

#define   DATATICKS_TO_COPY      UINT_MAX          // 복사 되는 틱의 수
#define   DATATICKS_TO_PRINT     20                // 저널에 보내진 틱의 수

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 표준 심볼 틱 데이터를 MqlTick 배열로 가져옵니다.
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- 표준 심볼의 첫 번째 및 마지막 수신 틱 시간을 출력합니다.
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 저널에 있는 표준 심볼의 마지막 틱 DATATICKS_TO_PRINT를 출력합니다.
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 종합 시세 창에 사용자 정의 심볼을 추가합니다.
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- 사용자 지정 심볼 가격 히스토리에 틱 배열 데이터를 추가합니다.
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", ar
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", add

//--- 사용자 정의 심볼 틱 데이터를 MqlTick 배열로 가져옵니다.
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- 사용자 정의 심볼의 첫 번째 및 마지막 수신 틱 시간을 출력합니다.
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- 저널에 있는 사용자 정의 심볼의 마지막 틱 DATATICKS_TO_PRINT를 출력합니다.
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       Sleep(16);
//--- Del 키를 누르면 생성된 사용자 정의 심볼 및 해당 데이터가 삭제됩니다.
       if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
         {
//--- 바 데이터 삭제
          int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
          if(deleted>0)
             PrintFormat("%d history bars of the custom symbol '%s' were successfully d

          //--- 틱 데이터 삭제
          deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
          if(deleted>0)
             PrintFormat("%d history ticks of the custom symbol '%s' were successfully

          //--- 심볼 삭제
          if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
             PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
          break;
         }
      }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   Requested 4294967295 ticks to download tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351183943 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.20 21:18:12.010

   The last 20 ticks for the standard symbol 'EURUSD':
     351183923th Tick: 2024.06.20 21:17:46.380 Bid=1.07124 (Info tick)
     351183924th Tick: 2024.06.20 21:17:47.779 Ask=1.07125 Bid=1.07125 (Info tick)
     351183925th Tick: 2024.06.20 21:17:48.584 Ask=1.07124 Bid=1.07124 (Info tick)
     351183926th Tick: 2024.06.20 21:17:49.481 Ask=1.07125 (Info tick)
     351183927th Tick: 2024.06.20 21:17:49.985 Ask=1.07122 Bid=1.07122 (Info tick)
     351183928th Tick: 2024.06.20 21:17:50.482 Ask=1.07124 Bid=1.07124 (Info tick)
     351183929th Tick: 2024.06.20 21:17:51.584 Ask=1.07123 Bid=1.07123 (Info tick)
     351183930th Tick: 2024.06.20 21:17:52.786 Ask=1.07124 Bid=1.07124 (Info tick)
     351183931th Tick: 2024.06.20 21:17:53.487 Ask=1.07125 Bid=1.07125 (Info tick)
     351183932th Tick: 2024.06.20 21:17:53.989 Ask=1.07126 Bid=1.07126 (Info tick)
     351183933th Tick: 2024.06.20 21:17:55.789 Ask=1.07125 Bid=1.07125 (Info tick)
     351183934th Tick: 2024.06.20 21:17:58.495 Ask=1.07126 Bid=1.07126 (Info tick)
     351183935th Tick: 2024.06.20 21:18:00.102 Bid=1.07126 (Info tick)
     351183936th Tick: 2024.06.20 21:18:00.698 Ask=1.07129 Bid=1.07129 (Info tick)
     351183937th Tick: 2024.06.20 21:18:03.699 Bid=1.07129 (Info tick)
     351183938th Tick: 2024.06.20 21:18:04.699 Ask=1.07128 Bid=1.07128 (Info tick)
     351183939th Tick: 2024.06.20 21:18:05.901 Ask=1.07129 Bid=1.07129 (Info tick)
     351183940th Tick: 2024.06.20 21:18:07.606 Ask=1.07128 Bid=1.07128 (Info tick)
     351183941th Tick: 2024.06.20 21:18:11.512 Ask=1.07127 Bid=1.07127 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
     351183942th Tick: 2024.06.20 21:18:12.010 Ask=1.07126 Bid=1.07126 (Info tick)
   ...
   Start of adding 351183943 ticks to the history of the custom symbol 'EURUSD.C'
   Added 351183943 ticks to the history of the custom symbol 'EURUSD.C' in 269890 ms
   ...
   Requested 351183943 ticks to download tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351183943 t
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.20 21:18:12.010

   The last 20 ticks for the custom symbol 'EURUSD.C':
     351183923th Tick: 2024.06.20 21:17:46.380 Ask=1.07124 Bid=1.07124 (Info tick)
     351183924th Tick: 2024.06.20 21:17:47.779 Ask=1.07125 Bid=1.07125 (Info tick)
     351183925th Tick: 2024.06.20 21:17:48.584 Ask=1.07124 Bid=1.07124 (Info tick)
     351183926th Tick: 2024.06.20 21:17:49.481 Ask=1.07125 Bid=1.07125 (Info tick)
     351183927th Tick: 2024.06.20 21:17:49.985 Ask=1.07122 Bid=1.07122 (Info tick)
     351183928th Tick: 2024.06.20 21:17:50.482 Ask=1.07124 Bid=1.07124 (Info tick)
     351183929th Tick: 2024.06.20 21:17:51.584 Ask=1.07123 Bid=1.07123 (Info tick)
     351183930th Tick: 2024.06.20 21:17:52.786 Ask=1.07124 Bid=1.07124 (Info tick)
     351183931th Tick: 2024.06.20 21:17:53.487 Ask=1.07125 Bid=1.07125 (Info tick)
     351183932th Tick: 2024.06.20 21:17:53.989 Ask=1.07126 Bid=1.07126 (Info tick)
     351183933th Tick: 2024.06.20 21:17:55.789 Ask=1.07125 Bid=1.07125 (Info tick)
     351183934th Tick: 2024.06.20 21:17:58.495 Ask=1.07126 Bid=1.07126 (Info tick)
     351183935th Tick: 2024.06.20 21:18:00.102 Ask=1.07126 Bid=1.07126 (Info tick)
     351183936th Tick: 2024.06.20 21:18:00.698 Ask=1.07129 Bid=1.07129 (Info tick)
     351183937th Tick: 2024.06.20 21:18:03.699 Ask=1.07129 Bid=1.07129 (Info tick)
     351183938th Tick: 2024.06.20 21:18:04.699 Ask=1.07128 Bid=1.07128 (Info tick)
     351183939th Tick: 2024.06.20 21:18:05.901 Ask=1.07129 Bid=1.07129 (Info tick)
     351183940th Tick: 2024.06.20 21:18:07.606 Ask=1.07128 Bid=1.07128 (Info tick)
     351183941th Tick: 2024.06.20 21:18:11.512 Ask=1.07127 Bid=1.07127 (Info tick)
     351183942th Tick: 2024.06.20 21:18:12.010 Ask=1.07126 Bid=1.07126 (Info tick)
   */
  }
//+------------------------------------------------------------------+
//| 사용자 지정 심볼 생성, 에러 코드 반환                                |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| 사용자 정의 심볼 삭제                                             |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastErro
      return(false);
     }
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 배열에서 지정된 틱 수를 가져옵니다.                                |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- 히스토리 데이터 로딩의 시작을 알립니다.
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- 틱 수신을 위해 3번 시도합니다.
   int attempts=0;
   while(attempts<3)
     {
      //--- 틱을 받기 전에 시작 시간을 측정합니다.
      uint start=GetTickCount();

      //--- 1970.01.01 00:00.001 이후의 틱 내역을 요청합니다(매개변수 =1ms)
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- 틱 수 및 소요 시간에 대한 정보 표시
         PrintFormat("The tick history for the '%s' symbol is received in the amount

         //--- 틱 히스토리가 동기화되면 오류 코드는 0과 같습니다. - 'true'를 반환합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms.
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- 카운트 시도
      attempts++;
      //--- 틱 데이터베이스 동기화가 끝날 때까지 기다리기 위한 1초 일시 중지
      Sleep(1000);
     }
//--- 3번의 시도에서 틱 복사에 실패했습니다.
   return(false);
  }
//+------------------------------------------------------------------+
//| 틱의 문자열 설명을 반환합니다.                                   |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES

//--- 틱 플래그 체크
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)    == TICK_FLAG_BUY);
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)   == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)    == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)    == TICK_FLAG_BID);
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)   == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- 우선 거래 플래그에 대한 체크를 확인하십시오(CustomTicksAdd()에는 플래그가 없습니다).
   if(buy_tick || sell_tick)
     {
      //--- 거래 틱에 대한 출력 형성
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- 정보 틱에 대한 출력을 약간 다르게 구성
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- 틱 설명 반환
```

© 2000-2025, MetaQuotes Ltd.

```
   return(desc);
  }
```

追加参照

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomTicksDelete

지정된 시간 간격 동안 사용자 지정 심볼의 가격 내역에서 모든 틱을 삭제.

```
int  CustomTicksDelete(
   const string      symbol,           // 심볼 이름
   long              from_msc,         // 시작일
   long              to_msc            // 종료일
   );
```

매개변수

심볼

[in] 사용자 지정 심볼 이름.

from_msc

[in] 가격 내역에서 제거할 지정된 범위 내의 첫 번째 틱 시간. 1970년 1월 1일 이후 시간(밀리초).

to_msc

[in] 가격 내역에서 제거할 지정된 범위 내의 마지막 틱 시간. 1970년 1월 1일 이후 시간(밀리초).

값 반환

Number of deleted ticks or -1 in case of an 오류가 발생한 경우 삭제된 틱 또는 -1.

예:

```
//+------------------------------------------------------------------+
//|                                            CustomTicksDelete.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // 사용자 지정 심볼명
#define   CUSTOM_SYMBOL_PATH     "Forex"           // 심볼을 생성할 그룹 명
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // 사용자 정의 심볼의 기반이 되는 심볼

#define   DATATICKS_TO_COPY      UINT_MAX          // 복사 되는 틱의 수
#define   DATATICKS_TO_DELETE    10                // number of deleted ticks
#define   DATATICKS_TO_PRINT     20                // 저널에 보내진 틱의 수

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 표준 심볼 틱 데이터를 MqlTick 배열로 가져옵니다.
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- 표준 심볼의 첫 번째 및 마지막 수신 틱 시간을 출력합니다.
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- 저널에 있는 표준 심볼의 마지막 틱 DATATICKS_TO_PRINT를 출력합니다.
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 종합 시세 창에 사용자 정의 심볼을 추가합니다.
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- 사용자 지정 심볼 가격 히스토리에 틱 배열 데이터를 추가합니다.
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", ar
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", add

//--- MqlTick 배열에 새로 추가된 사용자 정의 심볼 틱 데이터를 가져옵니다.
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- 사용자 정의 심볼의 첫 번째 및 마지막 수신 틱 시간을 출력합니다.
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
```

© 2000-2025, MetaQuotes Ltd.

```
                TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
                TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- 저널에 있는 사용자 정의 심볼의 마지막 틱 DATATICKS_TO_PRINT를 출력합니다.
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 히스토리에서 틱 범위를 삭제할 틱 시간을 밀리초 단위로 가져옵니다.
   long time_from=array[total-DATATICKS_TO_DELETE-1].time_msc;

//--- 삭제 DATATICKS_TO_DELETE 배열에서 사용자 정의 심볼의 마지막 틱 범위
   Print("...");
   start=GetTickCount();
   PrintFormat("Start deleting %u ticks in the history of the custom symbol '%s'", DAT
   int deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, time_from, array[total-2].time_ms
   PrintFormat("Deleted %u ticks in the history of the custom symbol '%s' in %u ms", d

//--- 새로 수정된 사용자 정의 심볼 틱 데이터를 MqlTick 배열로 가져옵니다.
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- 틱 범위가 제거된 사용자 정의 심볼의 첫 번째 및 마지막 틱 시간을 출력합니다.
   total=(int)array.Size();
   PrintFormat("Time of the first tick from the changed history: %s.%03u, Time of the
                TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
                TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- 저널에 있는 사용자 정의 심볼의 마지막 틱 DATATICKS_TO_PRINT를 출력합니다.
   PrintFormat("\nThe last %d ticks of custom symbol '%s' with modified history:", DAT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
//--- Del 키를 누르면 생성된 사용자 정의 심볼 및 해당 데이터가 삭제됩니다.
```

© 2000-2025, MetaQuotes Ltd.

```
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
//--- 바 데이터 삭제
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d

         //--- 틱 데이터 삭제
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- 심볼 삭제
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
        }
     }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   Requested 4294967295 ticks to get tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351199027 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 10:10:40.392

   The last 20 ticks for the standard symbol 'EURUSD':
     351199007th Tick: 2024.06.21 10:10:23.045 Bid=1.07032 (Info tick)
     351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
     351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 (Info tick)
     351199010th Tick: 2024.06.21 10:10:25.146 Bid=1.07032 (Info tick)
     351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
     351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
     351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
     351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
     351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
     351199016th Tick: 2024.06.21 10:10:30.258 Bid=1.07036 (Info tick)
     351199017th Tick: 2024.06.21 10:10:30.872 Ask=1.07035 Bid=1.07035 (Info tick)
     351199018th Tick: 2024.06.21 10:10:31.358 Ask=1.07036 (Info tick)
     351199019th Tick: 2024.06.21 10:10:31.859 Ask=1.07037 Bid=1.07037 (Info tick)
     351199020th Tick: 2024.06.21 10:10:32.377 Ask=1.07039 Bid=1.07039 (Info tick)
     351199021th Tick: 2024.06.21 10:10:32.962 Ask=1.0704 Bid=1.0704 (Info tick)
     351199022th Tick: 2024.06.21 10:10:33.961 Ask=1.07039 Bid=1.07039 (Info tick)
     351199023th Tick: 2024.06.21 10:10:34.667 Ask=1.0704 (Info tick)
     351199024th Tick: 2024.06.21 10:10:35.170 Bid=1.0704 (Info tick)
     351199025th Tick: 2024.06.21 10:10:38.266 Ask=1.07041 Bid=1.07041 (Info tick)
     351199026th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
   ...
   Start of adding 351199027 ticks to the history of the custom symbol 'EURUSD.C'
```

© 2000-2025, MetaQuotes Ltd.

```
Added 351199027 ticks to the history of the custom symbol 'EURUSD.C' in 261594 ms
...
Requested 351199027 ticks to get tick history for the symbol 'EURUSD.C'
The tick history for the 'EURUSD.C' symbol is received in the amount of 351199027 t
First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 10:10:40.392

The last 20 ticks for the custom symbol 'EURUSD.C':
  351199007th Tick: 2024.06.21 10:10:23.045 Ask=1.07032 Bid=1.07032 (Info tick)
  351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
  351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 Bid=1.07032 (Info tick)
  351199010th Tick: 2024.06.21 10:10:25.146 Ask=1.07032 Bid=1.07032 (Info tick)
  351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
  351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
  351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
  351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
  351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
  351199016th Tick: 2024.06.21 10:10:30.258 Ask=1.07036 Bid=1.07036 (Info tick)
  351199017th Tick: 2024.06.21 10:10:30.872 Ask=1.07035 Bid=1.07035 (Info tick)
  351199018th Tick: 2024.06.21 10:10:31.358 Ask=1.07036 Bid=1.07036 (Info tick)
  351199019th Tick: 2024.06.21 10:10:31.859 Ask=1.07037 Bid=1.07037 (Info tick)
  351199020th Tick: 2024.06.21 10:10:32.377 Ask=1.07039 Bid=1.07039 (Info tick)
  351199021th Tick: 2024.06.21 10:10:32.962 Ask=1.0704 Bid=1.0704 (Info tick)
  351199022th Tick: 2024.06.21 10:10:33.961 Ask=1.07039 Bid=1.07039 (Info tick)
  351199023th Tick: 2024.06.21 10:10:34.667 Ask=1.0704 Bid=1.0704 (Info tick)
  351199024th Tick: 2024.06.21 10:10:35.170 Ask=1.0704 Bid=1.0704 (Info tick)
  351199025th Tick: 2024.06.21 10:10:38.266 Ask=1.07041 Bid=1.07041 (Info tick)
  351199026th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
...
Start deleting 10 ticks in the history of the custom symbol 'EURUSD.C'
Deleted 10 ticks in the history of the custom symbol 'EURUSD.C' in 188 ms
...
Requested 351199027 ticks to get tick history for the symbol 'EURUSD.C'
The tick history for the 'EURUSD.C' symbol is received in the amount of 351199017 t
Time of the first tick from the changed history: 2011.12.19 00:00:08.000, Time of t

The last 20 ticks of custom symbol 'EURUSD.C' with modified history:
  351198997th Tick: 2024.06.21 10:10:14.935 Ask=1.07036 Bid=1.07036 (Info tick)
  351198998th Tick: 2024.06.21 10:10:15.533 Ask=1.07035 Bid=1.07035 (Info tick)
  351198999th Tick: 2024.06.21 10:10:17.736 Ask=1.07036 Bid=1.07036 (Info tick)
  351199000th Tick: 2024.06.21 10:10:18.540 Ask=1.07037 Bid=1.07037 (Info tick)
  351199001th Tick: 2024.06.21 10:10:19.046 Ask=1.07038 Bid=1.07038 (Info tick)
  351199002th Tick: 2024.06.21 10:10:19.542 Ask=1.07036 Bid=1.07036 (Info tick)
  351199003th Tick: 2024.06.21 10:10:20.041 Ask=1.07035 Bid=1.07035 (Info tick)
  351199004th Tick: 2024.06.21 10:10:21.041 Ask=1.07035 Bid=1.07035 (Info tick)
  351199005th Tick: 2024.06.21 10:10:21.544 Ask=1.07032 Bid=1.07032 (Info tick)
  351199006th Tick: 2024.06.21 10:10:22.344 Ask=1.07032 Bid=1.07032 (Info tick)
  351199007th Tick: 2024.06.21 10:10:23.045 Ask=1.07032 Bid=1.07032 (Info tick)
  351199008th Tick: 2024.06.21 10:10:24.045 Ask=1.07031 Bid=1.07031 (Info tick)
  351199009th Tick: 2024.06.21 10:10:24.545 Ask=1.07032 Bid=1.07032 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
      351199010th Tick: 2024.06.21 10:10:25.146 Ask=1.07032 Bid=1.07032 (Info tick)
      351199011th Tick: 2024.06.21 10:10:25.649 Ask=1.07037 Bid=1.07037 (Info tick)
      351199012th Tick: 2024.06.21 10:10:27.050 Ask=1.07036 Bid=1.07036 (Info tick)
      351199013th Tick: 2024.06.21 10:10:28.153 Ask=1.07039 Bid=1.07039 (Info tick)
      351199014th Tick: 2024.06.21 10:10:29.157 Ask=1.07037 Bid=1.07037 (Info tick)
      351199015th Tick: 2024.06.21 10:10:29.658 Ask=1.07036 Bid=1.07036 (Info tick)
      351199016th Tick: 2024.06.21 10:10:40.392 Ask=1.07042 Bid=1.07042 (Info tick)
    */
  }
//+------------------------------------------------------------------+
//| 사용자 지정 심볼 생성, 에러 코드 반환                              |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| 사용자 정의 심볼 삭제                                             |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 배열에서 지정된 틱 수를 가져옵니다.                                |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- 히스토리 데이터 로딩의 시작을 알립니다.
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- 틱 수신을 위해 3번 시도합니다.
   int attempts=0;
   while(attempts<3)
     {
      //--- 틱을 받기 전에 시작 시간을 측정합니다.
      uint start=GetTickCount();

      //--- 1970.01.01 00:00.001 이후의 틱 내역을 요청합니다(매개변수 =1ms)
      int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
      if(received!=-1)
        {
         //--- 틱 수 및 소요 시간에 대한 정보 표시
         PrintFormat("The tick history for the '%s' symbol is received in the amount o

         //--- 틱 히스토리가 동기화되면 오류 코드는 0과 같습니다. - 'true'를 반환합니다.
         if(GetLastError()==0)
            return(true);

         PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms.
                     symbol, received, GetTickCount()-start, GetLastError());
        }
      //--- 카운트 시도
      attempts++;
      //--- 틱 데이터베이스 동기화가 끝날 때까지 기다리기 위한 1초 일시 중지
      Sleep(1000);
     }
//--- 3번의 시도에서 틱 복사에 실패했습니다.
   return(false);
  }
//+------------------------------------------------------------------+
//| 틱의 문자열 설명을 반환합니다.                                     |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES

//--- 틱 플래그 체크
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)   == TICK_FLAG_BUY);
```

© 2000-2025, MetaQuotes Ltd.

```
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)  == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)   == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)   == TICK_FLAG_BID);
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)  == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- 우선 거래 플래그에 대한 체크를 확인하십시오(CustomTicksAdd()에는 플래그가 없습니다).
   if(buy_tick || sell_tick)
     {
      //--- 거래 틱에 대한 출력 형성
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
      desc += "(Trade tick)";
     }
   else
     {
      //--- 정보 틱에 대한 출력을 약간 다르게 구성
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- 틱 설명 반환
   return(desc);
  }
```

추가 참조

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomTicksReplace

지정된 시간 간격 내에 사용자 지정 심볼의 가격 기록을MqlTick 유형 배열의 데이터로 완전히 대체합니다.

```
int  CustomTicksReplace(
   const string     symbol,            // 심볼 이름
   long             from_msc,          // 시작일
   long             to_msc,            // 종료일
   const MqlTick&   ticks[],           // 사용자 지정 심볼에 적용할 데이터의 배열
   uint             count=WHOLE_ARRAY  // 사용할 ticks[] 배열 요소의 수
   );
```

매개변수

심볼

[in] 사용자 지정 심볼 이름.

from_msc

[in] 가격 내역에서 제거할 지정된 범위 내의 첫 번째 틱 시간. 1970년 1월 1일 이후 시간(밀리초).

to_msc

[in] 가격 내역에서 제거할 지정된 범위 내의 마지막 틱 시간. 1970년 1월 1일 이후 시간(밀리초).

ticks[]

[in] 시간에 따라 오름차순으로 정렬된 MqlTick 유형의 틱 데이터의 배열.

count=WHOLE_ARRAY

[in] 지정된 시간 간격 동안 교체에 사용할 *ticks[]* 배열 요소의 수. WHOLE_ARRAY 는 모든 *ticks[]* 배열 요소를 사용해야 함을 의미합니다.

값 반환

오류가 발생할 경우 업데이트된 틱 또는 -1.

주의

여러 틱은 종종 시세 스트림에서 최대 1밀리초까지 동일한 시간을 가질 수 있으므로 (정확한 틱 시간은 *time_msc* 필드에 저장됨, MqlTick 구조) CustomTicksReplace 기능은 시간별로*ticks[]* 배열 요소를 자동으로 정렬하지 않습니다. 따라서 틱 배열을 시간 오름차순으로 미리 정렬해야 합니다.

틱은 매일 즉, *to_msc* 또는 오류가 발생할 때까지 연속적으로 교체됩니다. 지정된 범위의 첫 번째 요일이 처리된 후 다음 요일 등이 처리됩니다. 틱 시간과 오름차순(비내림차순) 간의 불일치가 감지되는 즉시, 틱 교체가 현재 날짜에 중지됩니다. 현재 날짜(잘못된 틱의 순간)와 지정된 간격의 나머지 모든 일수는 변경되지 않은 상태에서 이전 날짜의 모든 체크 표시가 성공적으로 대체됩니다.

홀*ticks[]* 배열에 특정 날짜(일반적으로 임의의 시간 간격)의 틱 데이터가 없는 경우, *ticks[]*의 틱 데이터가 적용된 후 누락된 데이터에 해당하는 "홀(hole)"이 사용자 지정 심볼 내역에 나타납니다. 다시 말해, 누락된 틱이 있는 CustomTicksReplace 호출은 "홀(hole)" 간격이 있는 CustomTicksDelete를 호출하는 것처럼 틱의 일부를 삭제하는 것과 같습니다.

틱 데이터베이스에 지정된 시간 간격에 대한 데이터가 없는 경우 CustomTicksReplace은 ticks[] 배열에서 데이터베이스 틱에 추가됩니다.

© 2000-2025, MetaQuotes Ltd.

CustomTicksReplace 기능은 틱 데이터베이스와 직접 연동됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                          CustomTicksReplace.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#define   CUSTOM_SYMBOL_NAME     Symbol()+".C"     // 사용자 지정 심볼명
#define   CUSTOM_SYMBOL_PATH     "Forex"           // 심볼을 생성할 그룹 명
#define   CUSTOM_SYMBOL_ORIGIN   Symbol()          // 사용자 정의 심볼의 기반이 되는 심볼

#define   DATATICKS_TO_COPY      UINT_MAX          // 복사 되는 틱의 수
#define   DATATICKS_TO_PRINT     20                // 저널에 보내진 틱의 수

//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용자 정의 심볼을 생성할 때 오류 코드를 가져옵니다.
   int create=CreateCustomSymbol(CUSTOM_SYMBOL_NAME, CUSTOM_SYMBOL_PATH, CUSTOM_SYMBOL

//--- 오류 코드가 0(심볼 생성 성공)도 아니고 5304(심볼이 이미 생성됨)도 아닌 경우 - 그대로 둠
   if(create!=0 && create!=5304)
      return;

//--- 표준 심볼 틱 데이터를 MqlTick 배열로 가져옵니다.
   MqlTick array[]={};
   if(!GetTicksToArray(CUSTOM_SYMBOL_ORIGIN, DATATICKS_TO_COPY, array))
      return;

//--- 표준 심볼의 첫 번째 및 마지막 수신 틱 시간을 출력합니다.
   int total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time,TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- 저널에 있는 표준 심볼의 마지막 틱 DATATICKS_TO_PRINT를 출력합니다.
   PrintFormat("\nThe last %d ticks for the standard symbol '%s':", DATATICKS_TO_PRINT
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 종합 시세 창에 사용자 정의 심볼을 추가합니다.
   ResetLastError();
   if(!SymbolSelect(CUSTOM_SYMBOL_NAME, true))
     {
      Print("SymbolSelect() failed. Error ", GetLastError());
      return;
     }

//--- 사용자 지정 심볼 가격 히스토리에 틱 배열 데이터를 추가합니다.
   Print("...");
   uint start=GetTickCount();
   PrintFormat("Start of adding %u ticks to the history of the custom symbol '%s'", ar
   int added=CustomTicksAdd(CUSTOM_SYMBOL_NAME, array);
   PrintFormat("Added %u ticks to the history of the custom symbol '%s' in %u ms", add

//--- MqlTick 배열에 새로 추가된 사용자 정의 심볼 틱 데이터를 가져옵니다.
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- 사용자 정의 심볼의 첫 번째 및 마지막 수신 틱 시간을 출력합니다.
   total=(int)array.Size();
   PrintFormat("First tick time: %s.%03u, Last tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- 저널에 있는 사용자 정의 심볼의 마지막 틱 DATATICKS_TO_PRINT를 출력합니다.
   PrintFormat("\nThe last %d ticks for the custom symbol '%s':", DATATICKS_TO_PRINT,
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Tick: %s", i, GetTickDescription(array[i]));
     }

//--- 이제 Ask(Symbol) = 1.0 / Ask(Symbol), Bid(Symbol) = 1.0 / Bid(Symbol) 방정식을 사
   for(int i=0; i<total; i++)
     {
      array[i].ask = (array[i].ask !=0 ? 1.0 / array[i].ask : array[i].ask);
      array[i].bid = (array[i].bid !=0 ? 1.0 / array[i].bid : array[i].bid);
     }
   Print("\nNow the ticks are changed");

//--- 사용자 정의 심볼의 틱 기록을 수정된 틱 배열의 데이터로 바꿉니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   Print("...");
   start=GetTickCount();
   PrintFormat("Start replacing %u changed ticks in the history of the custom symbol
   int replaced=CustomTicksReplace(CUSTOM_SYMBOL_NAME, array[0].time_msc, array[total-
   PrintFormat("Replaced %u ticks in the history of the custom symbol '%s' in %u ms",

//--- 새로 교체된 사용자 정의 심볼 틱 데이터를 MqlTick 배열로 가져옵니다.
   Print("...");
   if(!GetTicksToArray(CUSTOM_SYMBOL_NAME, array.Size(), array))
      return;

//--- 사용자 정의 심볼의 수정된 첫 번째 및 마지막 수신된 틱의 시간을 인쇄합니다.
   total=(int)array.Size();
   PrintFormat("First changed tick time: %s.%03u, Last changed tick time: %s.%03u",
               TimeToString(array[0].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS), array
               TimeToString(array[total-1].time, TIME_DATE|TIME_MINUTES|TIME_SECONDS),

//--- 저널에 있는 사용자 정의 심볼의 마지막 수정 틱 DATATICKS_TO_PRINT를 출력합니다.
   PrintFormat("\nThe last %d changed ticks for the custom symbol '%s':", DATATICKS_TO
   for(int i=total-DATATICKS_TO_PRINT; i<total; i++)
     {
      if(i<0)
         continue;
      PrintFormat("  %dth Changed tick: %s", i, GetTickDescription(array[i]));
     }

//--- 차트 주석에 스크립트 종료 키에 대한 힌트를 표시합니다.
   Comment(StringFormat("Press 'Esc' to exit or 'Del' to delete the '%s' symbol and ex
//--- Esc 또는 Del 키를 눌러 무한 루프를 종료할 때까지 기다립니다.
   while(!IsStopped() && TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)==0)
     {
      Sleep(16);
//--- Del 키를 누르면 생성된 사용자 정의 심볼 및 해당 데이터가 삭제됩니다.
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_DELETE)<0)
        {
//--- 바 데이터 삭제
         int deleted=CustomRatesDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history bars of the custom symbol '%s' were successfully d

         //--- 틱 데이터 삭제
         deleted=CustomTicksDelete(CUSTOM_SYMBOL_NAME, 0, LONG_MAX);
         if(deleted>0)
            PrintFormat("%d history ticks of the custom symbol '%s' were successfully

         //--- 심볼 삭제
         if(DeleteCustomSymbol(CUSTOM_SYMBOL_NAME))
            PrintFormat("Custom symbol '%s' deleted successfully", CUSTOM_SYMBOL_NAME)
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      }
//--- 종료하기 전에 차트를 삭제
   Comment("");
   /*
   결과:
   Requested 4294967295 ticks to get tick history for the symbol 'EURUSD'
   The tick history for the 'EURUSD' symbol is received in the amount of 351195822 tic
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 08:39:03.113

   The last 20 ticks for the standard symbol 'EURUSD':
     351195802th Tick: 2024.06.21 08:38:10.076 Ask=1.07194 (Info tick)
     351195803th Tick: 2024.06.21 08:38:13.162 Ask=1.07195 (Info tick)
     351195804th Tick: 2024.06.21 08:38:13.872 Bid=1.07195 (Info tick)
     351195805th Tick: 2024.06.21 08:38:14.866 Ask=1.07194 Bid=1.07194 (Info tick)
     351195806th Tick: 2024.06.21 08:38:17.374 Bid=1.07194 (Info tick)
     351195807th Tick: 2024.06.21 08:38:18.883 Bid=1.07194 (Info tick)
     351195808th Tick: 2024.06.21 08:38:19.771 Bid=1.07194 (Info tick)
     351195809th Tick: 2024.06.21 08:38:20.873 Ask=1.07195 Bid=1.07195 (Info tick)
     351195810th Tick: 2024.06.21 08:38:22.278 Ask=1.07196 Bid=1.07196 (Info tick)
     351195811th Tick: 2024.06.21 08:38:22.775 Bid=1.07196 (Info tick)
     351195812th Tick: 2024.06.21 08:38:23.477 Bid=1.07196 (Info tick)
     351195813th Tick: 2024.06.21 08:38:38.194 Ask=1.07197 (Info tick)
     351195814th Tick: 2024.06.21 08:38:38.789 Ask=1.07196 (Info tick)
     351195815th Tick: 2024.06.21 08:38:39.290 Ask=1.07197 (Info tick)
     351195816th Tick: 2024.06.21 08:38:43.695 Ask=1.07196 (Info tick)
     351195817th Tick: 2024.06.21 08:38:52.203 Ask=1.07195 Bid=1.07195 (Info tick)
     351195818th Tick: 2024.06.21 08:38:55.105 Ask=1.07196 Bid=1.07196 (Info tick)
     351195819th Tick: 2024.06.21 08:38:57.607 Ask=1.07195 Bid=1.07195 (Info tick)
     351195820th Tick: 2024.06.21 08:39:00.512 Ask=1.07196 Bid=1.07196 (Info tick)
     351195821th Tick: 2024.06.21 08:39:03.113 Ask=1.07195 Bid=1.07195 (Info tick)
   ...
   Start of adding 351195822 ticks to the history of the custom symbol 'EURUSD.C'
   Added 351195822 ticks to the history of the custom symbol 'EURUSD.C' in 349407 ms
   ...
   Requested 351195822 ticks to get tick history for the symbol 'EURUSD.C'
   The tick history for the 'EURUSD.C' symbol is received in the amount of 351195822 t
   First tick time: 2011.12.19 00:00:08.000, Last tick time: 2024.06.21 08:39:03.113

   The last 20 ticks for the custom symbol 'EURUSD.C':
     351195802th Tick: 2024.06.21 08:38:10.076 Ask=1.07194 Bid=1.07194 (Info tick)
     351195803th Tick: 2024.06.21 08:38:13.162 Ask=1.07195 Bid=1.07195 (Info tick)
     351195804th Tick: 2024.06.21 08:38:13.872 Ask=1.07195 Bid=1.07195 (Info tick)
     351195805th Tick: 2024.06.21 08:38:14.866 Ask=1.07194 Bid=1.07194 (Info tick)
     351195806th Tick: 2024.06.21 08:38:17.374 Ask=1.07194 Bid=1.07194 (Info tick)
     351195807th Tick: 2024.06.21 08:38:18.883 Ask=1.07194 Bid=1.07194 (Info tick)
     351195808th Tick: 2024.06.21 08:38:19.771 Ask=1.07194 Bid=1.07194 (Info tick)
     351195809th Tick: 2024.06.21 08:38:20.873 Ask=1.07195 Bid=1.07195 (Info tick)
     351195810th Tick: 2024.06.21 08:38:22.278 Ask=1.07196 Bid=1.07196 (Info tick)
```

© 2000-2025, MetaQuotes Ltd.

```
      351195811th Tick: 2024.06.21 08:38:22.775 Ask=1.07196 Bid=1.07196 (Info tick)
      351195812th Tick: 2024.06.21 08:38:23.477 Ask=1.07196 Bid=1.07196 (Info tick)
      351195813th Tick: 2024.06.21 08:38:38.194 Ask=1.07197 Bid=1.07197 (Info tick)
      351195814th Tick: 2024.06.21 08:38:38.789 Ask=1.07196 Bid=1.07196 (Info tick)
      351195815th Tick: 2024.06.21 08:38:39.290 Ask=1.07197 Bid=1.07197 (Info tick)
      351195816th Tick: 2024.06.21 08:38:43.695 Ask=1.07196 Bid=1.07196 (Info tick)
      351195817th Tick: 2024.06.21 08:38:52.203 Ask=1.07195 Bid=1.07195 (Info tick)
      351195818th Tick: 2024.06.21 08:38:55.105 Ask=1.07196 Bid=1.07196 (Info tick)
      351195819th Tick: 2024.06.21 08:38:57.607 Ask=1.07195 Bid=1.07195 (Info tick)
      351195820th Tick: 2024.06.21 08:39:00.512 Ask=1.07196 Bid=1.07196 (Info tick)
      351195821th Tick: 2024.06.21 08:39:03.113 Ask=1.07195 Bid=1.07195 (Info tick)

    Now the ticks are changed
    ...
    Start replacing 351195822 changed ticks in the history of the custom symbol 'EURUSI
    Replaced 351195822 ticks in the history of the custom symbol 'EURUSD.C' in 452266 r
    ...
    Requested 351195822 ticks to get tick history for the symbol 'EURUSD.C'
    The tick history for the 'EURUSD.C' symbol is received in the amount of 351195822 t
    First changed tick time: 2011.12.19 00:00:08.000, Last changed tick time: 2024.06.2

    The last 20 changed ticks for the custom symbol 'EURUSD.C':
      351195802th Changed tick: 2024.06.21 08:38:10.076 Ask=0.93289 Bid=0.93289 (Info t
      351195803th Changed tick: 2024.06.21 08:38:13.162 Ask=0.93288 Bid=0.93288 (Info t
      351195804th Changed tick: 2024.06.21 08:38:13.872 Ask=0.93288 Bid=0.93288 (Info t
      351195805th Changed tick: 2024.06.21 08:38:14.866 Ask=0.93289 Bid=0.93289 (Info t
      351195806th Changed tick: 2024.06.21 08:38:17.374 Ask=0.93289 Bid=0.93289 (Info t
      351195807th Changed tick: 2024.06.21 08:38:18.883 Ask=0.93289 Bid=0.93289 (Info t
      351195808th Changed tick: 2024.06.21 08:38:19.771 Ask=0.93289 Bid=0.93289 (Info t
      351195809th Changed tick: 2024.06.21 08:38:20.873 Ask=0.93288 Bid=0.93288 (Info t
      351195810th Changed tick: 2024.06.21 08:38:22.278 Ask=0.93287 Bid=0.93287 (Info t
      351195811th Changed tick: 2024.06.21 08:38:22.775 Ask=0.93287 Bid=0.93287 (Info t
      351195812th Changed tick: 2024.06.21 08:38:23.477 Ask=0.93287 Bid=0.93287 (Info t
      351195813th Changed tick: 2024.06.21 08:38:38.194 Ask=0.93286 Bid=0.93286 (Info t
      351195814th Changed tick: 2024.06.21 08:38:38.789 Ask=0.93287 Bid=0.93287 (Info t
      351195815th Changed tick: 2024.06.21 08:38:39.290 Ask=0.93286 Bid=0.93286 (Info t
      351195816th Changed tick: 2024.06.21 08:38:43.695 Ask=0.93287 Bid=0.93287 (Info t
      351195817th Changed tick: 2024.06.21 08:38:52.203 Ask=0.93288 Bid=0.93288 (Info t
      351195818th Changed tick: 2024.06.21 08:38:55.105 Ask=0.93287 Bid=0.93287 (Info t
      351195819th Changed tick: 2024.06.21 08:38:57.607 Ask=0.93288 Bid=0.93288 (Info t
      351195820th Changed tick: 2024.06.21 08:39:00.512 Ask=0.93287 Bid=0.93287 (Info t
      351195821th Changed tick: 2024.06.21 08:39:03.113 Ask=0.93288 Bid=0.93288 (Info t
    */
  }
//+------------------------------------------------------------------+
//| 사용자 지정 심볼 생성, 에러 코드 반환                                |
//+------------------------------------------------------------------+
int CreateCustomSymbol(const string symbol_name, const string symbol_path, const strin
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 사용자 정의 심볼의 기반이 될 심볼명을 정의합니다.
   string origin=(symbol_origin==NULL ? Symbol() : symbol_origin);

//--- 사용자 정의 심볼 생성에 실패했고 오류 5304가 아닌 경우 저널에 이를 보고합니다.
   ResetLastError();
   int error=0;
   if(!CustomSymbolCreate(symbol_name, symbol_path, origin))
     {
      error=GetLastError();
      if(error!=5304)
         PrintFormat("CustomSymbolCreate(%s, %s, %s) failed. Error %d", symbol_name, s
     }
//--- 성공
   return(error);
  }
//+------------------------------------------------------------------+
//| 사용자 정의 심볼 삭제                                             |
//+------------------------------------------------------------------+
bool DeleteCustomSymbol(const string symbol_name)
  {
//--- 종합시세 창에서 심볼 숨기기
   ResetLastError();
   if(!SymbolSelect(symbol_name, false))
     {
      PrintFormat("SymbolSelect(%s, false) failed. Error %d", GetLastError());
      return(false);
     }

//--- 사용자 정의 심볼 삭제에 실패한 경우 이를 저널에 보고하고 'false'를 반환합니다.
   ResetLastError();
   if(!CustomSymbolDelete(symbol_name))
     {
      PrintFormat("CustomSymbolDelete(%s) failed. Error %d", symbol_name, GetLastError
      return(false);
     }
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 배열에서 지정된 틱 수를 가져옵니다.                                |
//+------------------------------------------------------------------+
bool GetTicksToArray(const string symbol, const uint count, MqlTick &array[])
  {
//--- 히스토리 데이터 로딩의 시작을 알립니다.
   PrintFormat("Requested %u ticks to get tick history for the symbol '%s'", count, sy

//--- 틱 수신을 위해 3번 시도합니다.
   int attempts=0;
   while(attempts<3)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 틱을 받기 전에 시작 시간을 측정합니다.
       uint start=GetTickCount();

       //--- 1970.01.01 00:00.001 이후의 틱 내역을 요청합니다(매개변수 =1ms)
       int received=CopyTicks(symbol, array, COPY_TICKS_ALL, 1, count);
       if(received!=-1)
         {
          //--- 틱 수 및 소요 시간에 대한 정보 표시
          PrintFormat("The tick history for the '%s' symbol is received in the amount o

          //--- 틱 히스토리가 동기화되면 오류 코드는 0과 같습니다. - 'true'를 반환합니다.
          if(GetLastError()==0)
             return(true);

          PrintFormat("%s: Ticks are not synchronized yet, %d ticks received for %d ms
                      symbol, received, GetTickCount()-start, GetLastError());
         }
       //--- 카운트 시도
       attempts++;
       //--- 틱 데이터베이스 동기화가 끝날 때까지 기다리기 위한 1초 일시 중지
       Sleep(1000);
      }
//--- 3번의 시도에서 틱 복사에 실패했습니다.
    return(false);
  }
//+------------------------------------------------------------------+
//| 틱의 문자열 설명을 반환합니다.                                   |
//+------------------------------------------------------------------+
string GetTickDescription(MqlTick &tick)
  {
   string desc=StringFormat("%s.%03u ", TimeToString(tick.time, TIME_DATE|TIME_MINUTES

//--- 틱 플래그 체크
   bool buy_tick   = ((tick.flags &TICK_FLAG_BUY)    == TICK_FLAG_BUY);
   bool sell_tick  = ((tick.flags &TICK_FLAG_SELL)   == TICK_FLAG_SELL);
   bool ask_tick   = ((tick.flags &TICK_FLAG_ASK)    == TICK_FLAG_ASK);
   bool bid_tick   = ((tick.flags &TICK_FLAG_BID)    == TICK_FLAG_BID);
   bool last_tick  = ((tick.flags &TICK_FLAG_LAST)   == TICK_FLAG_LAST);
   bool volume_tick= ((tick.flags &TICK_FLAG_VOLUME)== TICK_FLAG_VOLUME);

//--- 우선 거래 플래그에 대한 체크를 확인하십시오(CustomTicksAdd()에는 플래그가 없습니다).
   if(buy_tick || sell_tick)
     {
      //--- 거래 틱에 대한 출력 형성
      desc += (buy_tick ? StringFormat("Buy Tick: Last=%G Volume=%d ", tick.last, tick
      desc += (sell_tick? StringFormat("Sell Tick: Last=%G Volume=%d ",tick.last, tick
      desc += (ask_tick ? StringFormat("Ask=%G ", tick.ask) : "");
      desc += (bid_tick ? StringFormat("Bid=%G ", tick.ask) : "");
```

© 2000-2025, MetaQuotes Ltd.

```
      desc += "(Trade tick)";
     }
   else
     {
      //--- 정보 틱에 대한 출력을 약간 다르게 구성
      desc += (ask_tick   ? StringFormat("Ask=%G ",  tick.ask)    : "");
      desc += (bid_tick   ? StringFormat("Bid=%G ",  tick.ask)    : "");
      desc += (last_tick  ? StringFormat("Last=%G ", tick.last)   : "");
      desc += (volume_tick? StringFormat("Volume=%d ",tick.volume): "");
      desc += "(Info tick)";
     }
//--- 틱 설명 반환
   return(desc);
  }
```

추가 참조

CustomRatesDelete, CustomRatesUpdate, CustomTicksDelete, CopyTicks, CopyTicksRange

© 2000-2025, MetaQuotes Ltd.

CustomBookAdd

사용자 지정 심볼에 대한 시장 수준 상태를 전달.. 브로커 서버에서 가격이 도착하는 것처럼 Depth of Market(시장 깊이)를 브로드캐스팅 할 수 있음.

```
bool  CustomBookAdd(
   const string        symbol,            // 심볼 이름
   const MqlBookInfo&  books[]            // Depth of Market 요소를 설명하는 배열
   uint                count=WHOLE_ARRAY  // 사용할 요소의 수
   );
```

매개변수

심볼

[in] 사용자 지정 심볼 이름.

books[]

[in] Depth of Marekt(시장 깊이) 상태를 완벽하게 설명하는 일련의 MqlBookInfo 유형 데이터(전체 구매 및 판매 요청). 지난 시장 수준 상태는 이전 시장 수준을 완전히 대체.

count=WHOLE_ARRAY

[in] 함수에 전달할 'books' 배열 요소의 수 기본적으로 전체 배열이 사용됩니다.

값 반환

성공하면 true, 그렇지 않으면 false. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출.

주의

CustomBookAdd 기능은 플랫폼 인터페이스 또는 MarketBookAdd기능을 통해 Depth of Market(시장 깊이)을 여는 사용자 지정 심볼에만 사용할 수 있습니다.

Depth of Market을 삽입할 때, 심볼의 입찰 및 요청 가격은 업데이트되지 않습니다. CustomTicksAdd를 사용하여 최적의 가격 변경을 제어하고 틱에 넣어야합니다.

이 기능은 전송된 데이터의 정확성을 확인합니다: 유형, 가격 및 볼륨은 각 요소에 대해 표시되어야 합니다 또한 MqlBookInfo.volume 및 MqlBookInfo.volume_real은 0 또는 음수가 아니어야 합니다; 만약 두 볼륨이 모두 음수이면 오류로 간주됩니다.볼륨 중 하나 또는 둘 모두를 지정할 수 있습니다: 지정된 볼륨이나 양수인 볼륨이 사용됩니다.

```
volume=-1 && volume_real=2 — volume_real=2 가 사용될 것입니다
a volume=3 && volume_real=0 — volume=3 가 사용될 것입니다.
```

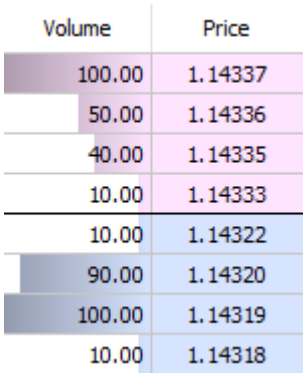
'books' 배열의 MqlBookInfo 요소 순서는 중요하지 않습니다. 터미널은 데이터를 저장할 때 스스로 가격을 기준으로 데이터를 정렬합니다.

데이터를 저장할 때 수신인 사용자 지정 심볼의 "Book depth" (SYMBOL_TICKS_BOOKDEPTH) 매개변수를 확인합니다. 전달된 Depth of Market에서 판매 요청 수가 이 값을 초과하면 초과 레벨이 폐기됩니다. 구매 요청도 마찬가지입니다.

'books' 배열의 샘플 채우기:

© 2000-2025, MetaQuotes Ltd.

Depth of Market 상태	Filling books[]

	books[0].type=BOOK_TYPE_SELL;
	books[0].price=1.14337;
	books[0].volume=100;
	books[1].type=BOOK_TYPE_SELL;
	books[1].price=1.14330;
	books[1].volume=50;
	books[2].type=BOOK_TYPE_SELL;
	books[2].price=1.14335;
	books[2].volume=40;
	books[3].type=BOOK_TYPE_SELL;
	books[3].price=1.14333;
	books[3].volume=10;
	books[4].type=BOOK_TYPE_BUY;
	books[4].price=1.14322;
	books[4].volume=10;
	books[5].type=BOOK_TYPE_BUY;
	books[5].price=1.14320;
	books[5].volume=90;
	books[6].type=BOOK_TYPE_BUY;
	books[6].price=1.14319;
	books[6].volume=100;
	books[7].type=BOOK_TYPE_BUY;
	books[7].price=1.14318;
	books[7].volume=10;

예를 들어:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---다음에서 데이터를 검색할 수 있는 Depth of Market(시장 깊이)를 활성화
   MarketBookAdd(Symbol());
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 해제 기능                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| 틱 기능                                                          |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   MqlTick ticks[];
   ArrayResize(ticks,1);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 통상의 현시가를 통상의 현시세로 복사
   if(SymbolInfoTick(Symbol(),ticks[0]))
     {
      string symbol_name=Symbol()+".SYN";
      CustomTicksAdd(symbol_name,ticks);
     }
  }
//+------------------------------------------------------------------+
//| 장부 기능                                                        |
//+------------------------------------------------------------------+
void OnBookEvent(const string &book_symbol)
  {
//--- 현재 Depth of Market(시장 깊이) 상태를 공통 심볼에서 사용자 지정 심볼로 복사
   if(book_symbol==Symbol())
     {
      MqlBookInfo book_array[];
      if(MarketBookGet(Symbol(),book_array))
        {
         string symbol_name=Symbol()+".SYN";
         CustomBookAdd(symbol_name,book_array);
        }
     }
  }
//+------------------------------------------------------------------+
```

추가 참조

MarketBookAdd, CustomTicksAdd, OnBookEvent

© 2000-2025, MetaQuotes Ltd.

차트 작업

차트 속성 설정 기능(ChartSetInteger, ChartSetDouble, ChartSetString)은 비동기식으로 업데이트 명령을 차트에 보내는 데 사용됩니다. 이러한 기능이 성공적으로 실행되면 명령은 차트 이벤트의 공통 대기열에 포함됩니다. 차트 속성 변경은 이 차트의 이벤트 대기열 처리와 함께 구현됩니다.

따라서 비동기 함수 호출한 후 차트를 즉시 업데이트하지 마십시오. 차트 모양 및 속성을 강제 업데이트하려면 ChartRedraw() 기능을 사용하십시오.

기능	액션
ChartApplyTemplate	지정된 파일의 특정 탬플릿을 차트에 적용
ChartSaveTemplate	지정된 이름의 템플릿에 현재 차트 설정을 저장
ChartWindowFind	지표가 그려진 하위창의 숫자를 반환
ChartTimePriceToXY	시간/가격 표시에서 X 및 Y 좌표로 차트의 좌표를 변환
ChartXYToTimePrice	차트의 X 및 Y 좌표를 시간 및 가격 값으로 변환
ChartOpen	지정된 심볼 및 기간으로 새 차트를 열기
ChartClose	지정된 차트를 닫기
ChartFirst	클라이언트 터미널의 첫 번째 차트의 ID를 받환
ChartNext	지정한 차트 옆에 있는 차트의 차트 ID를 반환
ChartSymbol	지정된 차트의 심볼명 반환
ChartPeriod	지정된 차트의 기간 값을 반환
ChartRedraw	지정된 차트의 강제 재작성 요청
ChartSetDouble	지정된 차트의 해당 속성에 대한 double 값을 설정
ChartSetInteger	지정된 차트의 해당 속성에 대한 정수 값 (datetime, int, color, bool or char) 을 설정
ChartSetString	지정된 차트의 해당 속성에 대한 string 값을 설정
ChartGetDouble	지정된 차트의 double 값 속성을 반환
ChartGetInteger	지정된 차트의 정수 값 속성을 반환
ChartGetString	지정한 차트의 string 값 속성을 반환
ChartNavigate	차트에서 지정된 포지션을 기준으로 지정된 막대 수만큼 지정된 차트의 이동을 수행
ChartID	현재 차트의 ID를 반화
ChartIndicatorAdd	지정된 핸들이 있는 지표를 지정된 차트 창에 추가
ChartIndicatorDelete	지정된 차트 창에서 지정된 이름의 지표를 제거
ChartIndicatorGet	지정된 차트 창에 지정된 단축명을 가진 지표의 핸들을 반환

© 2000-2025, MetaQuotes Ltd.

기능	액션
ChartIndicatorName	지정된 차트 창의 지표 목록에 있는 수를 기준으로 지표의 단축명을 반환
ChartIndicatorsTotal	지정된 차트 창에 적용된 모든 지표의 수를 반환
ChartWindowOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 하위 창의 숫자(인덱스)를 반환
ChartPriceOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트의 가격 좌표를 반환
ChartTimeOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인의 시간 좌표를 반환
ChartXOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트의 X좌표를 반환
ChartYOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트의 Y좌표를 반환
ChartSetSymbolPeriod	지정된 차트의 심볼 값과 기간을 변경
ChartScreenShot	지정된 확장자에 따라 GIF, PNG 또는 BMP 형식의 현재 상태 차트를 스크린샷을 제공

© 2000-2025, MetaQuotes Ltd.

ChartApplyTemplate

지정된 파일의 특정 템플릿을 차트에 적용. 이 명령은 차트 메시지 대기열에 추가되며 이전 모든 명령을 처리한 후 실행됩니다.

```
bool  ChartApplyTemplate(
   long          chart_id,     // 차트 ID
   const string  filename      // 템플릿 파일명
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

파일명

[in] 템플릿이 포함된 파일의 이름.

값 반환

명령이 차트 대기열에 추가된 경우 true를 반환, 그렇지 않은 경우 false를 반환. 오류에 대한 정보를 가져오기 위해, GetLastError() 함수 호출.

주의

엑스퍼트 어드바이저는 언로드되며, 첨부된 차트에 새 탬플릿을 성공적으로 로드하는 경우 작업을 계속할 수 없습니다.

차트에 템플릿을 적용할 때 보안상의 이유로 거래 권한이 제한될 수 있습니다.

> ChartApplyTemplate() 함수를 사용하여 템플릿을 적용하여 시작한 엑스퍼트 어드바이저에 대한 실시간 거래 권한을 확장할 수 없습니다. function.

mql5 프로그램 호출 ChartApplyTemplate() 함수에 거래 권한이 없는 경우, 템플릿을 통해 시작된 엑스퍼트 어드바이저도 템플릿 설정에 관계없이 거래할 수 없습니다.

mql5 프로그램 호출 ChartApplyTemplate() 함수에 거래 권한이 있는 반면 템플릿 설정에 해당 권한이 없는 경우, 템플릿을 통해 시작된 엑스퍼트 어드바이저는 거래할 수 없습니다.

템플릿 사용

MQL5 언어의 리소스를 사용하면 ChartSetInteger() 함수를 사용하여 색상을 포함한 여러 차트의 속성을 설정할 수 있습니다:

- 차트 배경색상;
- 축, 스케일 및 OHLC 라인 색상;
- 그리드 색상;
- 볼륨 및 포지션 시작 레벨 색상;
- 상단 막대 색상, 그림자와 강세 캔들스틱의 엣지;
- 하단 막대 색상, 그림자와 약세 캔들스틱의 엣지;
- 차트라인 및 도지 캔들스틱의 색상;
- 강세 캔들스틱 바디 색상;

© 2000-2025, MetaQuotes Ltd.

- 약세 캔들스택 바디 색상;
- 입찰 가격 라인 색상;
- 요청 가격 라인 색상;
- 마지막 거래 가격의 라인 색상;
- 정지 주문 수준의 색상(손실 중지 및 이익 획득).

또한, 차트에 여러 그래픽 개체 및 지표가 있을 수 있습니다. 필요한 모든 지표가 포함된 차트를 한 번 설정한 다음 템플릿으로 저장할 수 있습니다. 이러한 테플릿은 모든 차트에 적용할 수 있습니다.

ChartApplyTemplate() 기능은 이전에 저장된 템플릿을 사용하기 위한 것으로, 모든 mql5 프로그램에서 사용할 수 있습니다. 템플릿을 저장하는 파일의 경로는 ChartApplyTemplate()에 대한 두 번째 변수로 전달됩니다. 템플릿 파일은 다음 규칙에 따라 검색됩니다:

- 백슬래시 "₩ 구분기호 ("₩₩)로 작성된 경우, _terminal_data_directory₩MQL5 경로를 기준으로 템플릿이 검색됩니다,
- 백슬래시가 없는 경우 ChartApplyTemplate()가 호출되는 실행가능한 EX5파일에 대해 템플릿이 검색됩니다;
- 처음 두 변수에서 템플릿을 찾을 수 없는 경우 terminal_directory₩Profiles₩Templates₩폴더에서 검색이 수행됩니다.

여기서 terminal_directory는 MetaTrader 5 클라이언트 터미널이 실행중인 폴더이며, terminal_data_directory는 편집 가능한 파일이 저장되는 폴더이고, 운영 체제, 사용자 이름 및 컴퓨터의 보안 설정에 따라 위치가 달라집니다. 일반적으로 서로 다른 폴더지만, 경우에 따라 일치할 수 있습니다.

terminal_data_directory 및 terminal_directory 폴더 위치는TerminalInfoString() 함수를 사용하여 얻을 수 있습니다.

```
//--- 터미널이 시작되는 디렉토리
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("터미널 디렉토리:",terminal_path);
//--- EA 및 지표가 있는 MQL5 폴더가 위치한 터미널 데이터 디렉토리
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("터미널 데이터 디렉토리:",terminal_data_path);
```

예를 들어:

```
//--- terminal_data_directory₩MQL5₩에서 템플릿 검색
ChartApplyTemplate(0,"\\first_template.tpl"))
//--- directory_of_EX5_file₩에서 템플릿을 검색한 다음, terminal_data_directory₩Profiles₩T
ChartApplyTemplate(0,"second_template.tpl"))
//--- directory_of_EX5_file₩My_templates₩에서 템플릿을 검색한 다음, terminal_directory₩Pr
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

템플릿은 리소스가 아니므로 실행 가능한 EX5 파일에 포함할 수 없습니다.

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ₩MQL5₩Files에 있는 템플릿 적용 예시
```

© 2000-2025, MetaQuotes Ltd.

```
   if(FileIsExist("my_template.tpl"))
     {
      Print("my_template.tpl 파일을 'Files' 에서 찾았습니다");
      //--- 템플릿 적용
      if(ChartApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
         Print("템플릿 'my_template.tpl' 가 성공적으로 적용되었습니다");
         //--- 차트 재작성
         ChartRedraw();
        }
      else
         Print("'my_template.tpl' 을 적용하지 못했습니다, 오류 코드 ",GetLastError());
     }
   else
     {
      Print("File 'my_template.tpl' 파일을 찾지 못했습니다 "
            +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
     }
  }
```

추가 참조

리소스

© 2000-2025, MetaQuotes Ltd.

ChartSaveTemplate

지정된 이름의 템플릿에 현재 차트 설정을 저장

```
bool  ChartSaveTemplate(
   long          chart_id,     // 차트 ID
   const string  filename      // 템플릿을 저장할 파일명
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

파일명

[in] 템플릿을 저장할 파일명. ".tpl" 확장자는 파일명에 자동으로 추가되므로 지정할 필요가 없습니다. 템플릿은 **data_folder**₩Profiles₩Templates₩에 저장되며, 터미널에서 수동 애플리케이션에 사용할 수 있습니다. 동일한 파일 이름을 가진 템플릿이 이미 있으면 이 파일의 내용을 덮어씁니다.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수 호출.

주의

템플릿을 사용하여 적용된 모든 지표 및 그래픽 개체와 함께 차트 설정을 저장한 다음 다른 차트에 적용할 수 있습니다.

예를 들어:

```
//+------------------------------------------------------------------+
//|                                    Test_ChartSaveTemplate.mq5 |
//|                              Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- 매개변수 입력
input string               symbol="GBPUSD";  // 새 차트의 심볼
input ENUM_TIMEFRAMES      period=PERIOD_H3; // 새 차트의 시간프레임
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 차트에 지표를 먼저 첨부
   int handle;
//--- 사용할 지표 준비
   if(!PrepareZigzag(NULL,0,handle)) return; // 실패하여 종료
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 현재 차트에 지표를 첨부하지만 별도의 창에 지표 첨부.
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("handl e=%d로 %s/%s 지표를 차트에 연결하는데 실패하였습니다. 오류 코드 %d",
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- 프로그램 작업 종료
      return;
     }
//--- 지표를 표시하기 위해 차트를 새로고침
   ChartRedraw();
//--- 마지막 두 지그재그 프랙쳐 찾기
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("마지막 두 지그재그 프랙쳐를 발견하는 데 실패하였습니다!");
      //--- 프로그램 작업 종료
      return;
     }
//--- 이제 표준 편차 채널을 첨부
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("%s 개체를 생성하는 데 실패하였습니다. 오류 코드 %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- 두 번째 포인트를 정의하는 채널이 생성되었습니다.
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- 채널에 대한 도구 설명 텍스트 설정
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
      //--- 차트 새로고침
      ChartRedraw();
     }
//--- 결과를 템플릿에 저장
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- 새 차트를 열고 저장된 템플릿을 적용
   long new_chart=ChartOpen(symbol,period);
   //--- 그래픽 개체에 대한 도구 설명 사용
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
      //--- 차트에 저장된 템플릿 적용
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    Sleep(10000);
   }
//+------------------------------------------------------------------+
//| 지그재그 핸들 생성 및 데이터 준비 보장                           |
//+------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
   {
    ResetLastError();
//--- 지그재그 지표는 terminal_data_folder\MQL5\Examples에 위치해야 합니다
    h=iCustom(sym,tf,"Examples\\Zigzag");
    if(h==INVALID_HANDLE)
      {
       PrintFormat("%s: 지그재그 지표의 핸들을 생성하는 데 실패하였습니다. 오류 코드 %d",
                   __FUNCTION__,GetLastError());
       return false;
      }
//--- 지표 핸들을 생성할 때 값을 계산하는 데 시간이 필요합니다
    int k=0; // 지표 계산을 대기하는 시도 횟수
//--- 루프에서 계산을 기다렸다가 계산이 아직 준비되지 않은 경우 50밀리초로 일시 중지
    while(BarsCalculated(h)<=0)
      {
       k++;
       //--- 시도 횟수 표시
       PrintFormat("%s: k=%d",__FUNCTION__,k);
       //--- 50밀리초 동안 기다렸다가 지표가 계산될 때까지 기다리십시오
       Sleep(50);
       //--- 만약 100번 이상 시도한다면, 뭔가 잘못된 것입니다
       if(k>100)
         {
          //--- 문제 보고
          PrintFormat("%d 시도에 대한 지표를 계산에 실패하였습니다!");
          //--- 프로그램 작업 종료
          return false;
         }
      }
//--- 모든 것이 준비되고 지표가 생성되며 값이 계산됩니다.
    return true;
   }
//+------------------------------------------------------------------+
//| 마지막 2개의 지그재그 프랙쳐 검색 및 배열에 배치                 |
//+------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
   {
    double values[];          // 지그재그 값에 대한 배열
    datetime times[];         // 시간을 가져오기 위한 배열
    int size=100;             // 배열의 크기
    ResetLastError();
//--- 지표의 마지막 100개의 값을 복사
```

© 2000-2025, MetaQuotes Ltd.

```
   int copied=CopyBuffer(handle,0,0,size,values);
//--- 복사한 값 수 확인
   if(copied<100)
     {
      PrintFormat("%s: handle=%d를 사용하여 지표의 %d 값을 복사에 실패했습니다. 오류 코드 %d
                  __FUNCTION__,size,handle,GetLastError());
      return false;
     }
//--- 시계열에서와 같이 배열에 대한 액세스 순서 정의
   ArraySetAsSeries(values,true);
//--- 여기에 프랙쳐가 발견된 막대의 수를 쓰기
   int positions[];
//--- 배열 크기 설정
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- 카운터
   int i=0,k=0;
//--- 프랙쳐 검색을 시작
   while(i<100)
     {
      double v=values[i];
      //--- 비어있는 값에는 관심이 없습니다
      if(v!=0.0)
        {
         //--- 막대 숫자를 기억하기
         positions[k]=i;
         //--- 프랙쳐의 지그재그 값을 기억하기
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- 카운터 증가
         k++;
         //--- 두 프랙처가 발견되면, 루프 중단
         if(k>2) break;
        }
      i++;
     }
//--- 시계열에서와 같이 배열에 대한 액세스 순서 정의
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: CopyTime()에서 %d 값을 복사에 실패했습니다. 오류 코드 %d",
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- 막대가 열린 시간(마지막 2회 프랙처가 발생한 시간)을 열기
   get_times[0]=times[positions[1]];// 마지막 값 하나만 첫 번째 프랙처로 기록됩니다
   get_times[1]=times[positions[2]];// 끝에서 세 번째 값은 두 번째 프랙처입니다
   PrintFormat("%s: first=%s,  second=%s",__FUNCTION__,TimeToString(get_times[1]),Time
//--- 성공
   return true;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

추가 참조

ChartApplyTemplate(), Resources

© 2000-2025, MetaQuotes Ltd.

ChartWindowFind

이 함수는 지표가 그려진 하위 창의 번호를 반환. 함수의 2가지 변수가 있습니다.

1. 함수는 표시된 차트에서 지표의 지정된 "단축명"(단축명은 하위 창의 왼쪽 상단 부분에 표시됨)을 사용하여 하위 창을 검색하고 성공한 경우 하위 창 번호를 반환합니다.

```
int  ChartWindowFind(
   long     chart_id,                  // 차트 식별자
   string   indicator_shortname        // 지표 단축명은 다음을 보십시오 INDICATOR_SHORTNA
```

2. 사용자 지정 지표에서 함수를 호출해야 합니다. 지표가 작동하는 하위 창의 번호를 반환합니다.

```
int  ChartWindowFind();
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 나타냅니다.

indicator_shortname

[in] 지표의 단축명

값 반환

성공 시 하위 창의 번호 실패 시 함수는 -1을 반환.

주의

매개변수가 없는 함수의 두 번째 변수를 스크립트 또는 엑스퍼트 어드바이저에서 호출하면 함수는 -1을 반환합니다.

지표의 단축명과 파일 이름명을 혼동하시 마십시오. 이 이름은 iCustom() 및 IndicatorCreate() 함수를 사용하여 지표를 만들 때 지정됩니다. 지표의 단축명이 명시적으로 설정되지 않은 경우 지표의 소스코드를 포함하는 파일명이 컴파일 중에 지정됩니다.

INDICATOR_SHORTNAME 속성에 IndicatorSetString() 기능을 사용하여 기록되는 지표의 단축명을 올바르게 구성하는 것이 중요합니다. ChartIndicatorDelete() 기능의 차트에서 지표는 단축명으로 식별되는 단축명에 지표의 입력 매개변수 값을 포함할 것을 권장합니다.

예를 들어:

```
#property script_show_inputs
//--- 매개변수 입력
input string   shortname="MACD(12,26,9)";
//+------------------------------------------------------------------+
//| 이 지표가 있는 차트 창의 번호를 반환                                  |
//+------------------------------------------------------------------+
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
  {
   int window=-1;
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 이 함수는 지표에서 호출되며, 이름은 필요하지 않습니다
      window=ChartWindowFind();
     }
   else
     {
      //--- 이 함수는 엑스퍼트 어드바이저 또는 스크립트에서 호출됩니다
      window=ChartWindowFind(0,short_name);
      if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
     }
//---
   return(window);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int window=GetIndicatorSubWindowNumber(0,shortname);
   if(window!=-1)
      Print("Indicator "+shortname+" is in the window #"+(string)window);
   else
      Print("지표 "+shortname+" 를 발견하지 못했습니다. wi ndow = "+(string)window);
  }
```

추가 참조

ObjectCreate(), ObjectFind()

© 2000-2025, MetaQuotes Ltd.

ChartTimePriceToXY

시간/가격 표시에서 X 및 Y 좌표로 차트의 좌표에 반환.

```
bool  ChartTimePriceToXY(
   long           chart_id,      // 차트 ID
   int            sub_window,    // 하위 창의 번호
   datetime       time,          // 차트 상의 시간
   double         price,         // 차트 상의 가격
   int&           x,             // 차트의 시간에 대한 X 좌표
   int&           y              //차트의 시간에 대한 Y 좌표
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

sub_window

[in] 차트 하위 창의 번호. 0은 주 차트 창을 의미합니다.

시간

[in] X축을 따라 픽셀 단위로 수신되는 차트의 시간 값입니다. 원점은 주 차트의 창 왼쪽 위에 있습니다.

가격

[in] Y 축을 따라 픽셀 단위로 표시되는 차트의 가격 값. 원점은 주 차트의 창 왼쪽 위에 있습니다.

x

[out] X로의 시간 변환이 수신될 변수.

y

[out] Y로의 가격 변환이 수신될 변수.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수 호출.

예:

```
#define   BAR_NUMBER    0  // 가격과 시간을 가져오는 바의 번호

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- BAR_NUMBER 인덱스로 단일 바의 데이터를 복사합니다.
   MqlRates rates[]={};
   if(CopyRates(_Symbol, _Period, BAR_NUMBER, 1, rates)!=1)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("CopyRates() failed for bar %d. Error %d", BAR_NUMBER, GetLastError
      return;
     }

//--- 획득한 가격과 시간을 차트 픽셀 좌표로 변환합니다.
   int x=0, y=0;
   ResetLastError();
   if(!ChartTimePriceToXY(ChartID(), 0, rates[0].time, rates[0].close, x, y))
     {
      Print("ChartTimePriceToXY() failed. Error ", GetLastError());
      return;
     }

//--- 획득한 결과를 저널에 출력합니다.
   PrintFormat("For bar[%d] with opening time %s and price %.*f, the chart coordinates

   /*
   결과:
   For bar[0] with opening time 2024.08.09 15:06 and price 1.27378, the chart coordina
   */
  }
```

추가 참조

ChartXYToTimePrice()

© 2000-2025, MetaQuotes Ltd.

ChartXYToTimePrice

차트의 X 및 Y 좌표를 시간 및 가격 값으로 반환.

```
bool  ChartXYToTimePrice(
   long           chart_id,     // 차트 ID
   int            x,            // 차트의 X좌표
   int            y,            // 차트의 Y좌표
   int&           sub_window,   // 하위 창의 번호
   datetime&      time,         // 차트의 시간
   double&        price         // 차트의 가격
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

x

[in] X좌표.

y

[in] Y좌표.

sub_window

[out] 차트 하위 창 번호를 기록할 변수. 0은 주 차트 창을 의미합니다.

시간

[out] X축을 따라 픽셀 단위로 수신될 차트의 시간 값. 원점은 주 차트의 창 왼쪽 위에 있습니다.

가격

[out] Y축을 따라 픽셀 단위로 표시되는 차트의 가격 값. 원점은 주 차트의 창 왼쪽 위에 있습니다.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수 호출.

예를 들어:

```
//+------------------------------------------------------------------+
//| ChartEvent 기능                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- 차트에 이벤트 매개변수 표시
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam," sparam=",spar
//--- 차트의 마우스 클릭 이벤트인 경우
   if(id==CHARTEVENT_CLICK)
```

© 2000-2025, MetaQuotes Ltd.

```
    {
     //--- 변수를 준비
     int      x     =(int)lparam;
     int      y     =(int)dparam;
     datetime dt    =0;
     double   price =0;
     int      window=0;
     //--- 날짜/시간 단위로 X 및 Y 좌표 변환
     if(ChartXYToTimePrice(0,x,y,window,dt,price))
       {
        PrintFormat("Window=%d X=%d  Y=%d  =>  Time=%s  Price=%G",window,x,y,TimeToSt
        //--- 역방향 변환 수행: (X,Y) => (Time,Price)
        if(ChartTimePriceToXY(0,window,dt,price,x,y))
           PrintFormat("Time=%s  Price=%G  =>  X=%d  Y=%d",TimeToString(dt),price,x,y
        else
           Print("ChartTimePriceToXY return error code: ",GetLastError());
        //--- 열 삭제
        ObjectDelete(0,"V Line");
        ObjectDelete(0,"H Line");
        //--- 크로스헤어의 가로 및 세로열을 작성
        ObjectCreate(0,"H Line",OBJ_HLINE,window,dt,price);
        ObjectCreate(0,"V Line",OBJ_VLINE,window,dt,price);
        ChartRedraw(0);
       }
     else
        Print("ChartXYToTimePrice return error code: ",GetLastError());
     Print("+------------------------------------------------------------------+");
    }
  }
```

추가 참조

ChartTimePriceToXY()

© 2000-2025, MetaQuotes Ltd.

ChartOpen

지정된 심볼과 기간으로 새 차트를 열기

```
long  ChartOpen(
   string           symbol,     // 심볼 이름
   ENUM_TIMEFRAMES  period      // 기간
   );
```

매개변수

심볼

[in] 차트 심볼. NULL 은 현재 차트의 심볼을 의미합니다 (엑스퍼트 어드바이저가 첨부됨).

기간

[in] 차트 기간 (시간프레임). ENUM_TIMEFRAMES 값 중 하나일 수 있습니다. 0은 현재 차트 기간을 의미합니다.

값 반환

성공하면 열린 차트 ID를 반환합니다. 그렇지 않으면 0을 반환합니다.

주의

터미널에서 동시에 열려 있는 최대 차트 수는 CHARTS_MAX 값을 초과할 수 없습니다.

예:

```
#define CHART_SYMBOL  NULL
#define CHART_PERIOD  PERIOD_CURRENT

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 새로운 차트 심볼과 차트 주기를 설정
   string symbol=CHART_SYMBOL;
   if(symbol==NULL)
      symbol=Symbol();
   ENUM_TIMEFRAMES timeframe = (CHART_PERIOD==PERIOD_CURRENT ? Period() : CHART_PERIOI

//--- 지정된 심볼과 기간을 사용하여 새 차트를 엽니다.
   long chart_id=ChartOpen(symbol, timeframe);
   if(chart_id==0)
     {
      Print("ChartOpen() failed. Error ", GetLastError());
      return;
     }

//--- 저널에서 오픈 차트 매개변수를 인쇄합니다.
   PrintFormat("A new chart of the %s symbol has been opened with a period of %s and (
```

© 2000-2025, MetaQuotes Ltd.

```
                symbol, StringSubstr(EnumToString(timeframe), 7), chart_id);
  /*
  결과:
  A new chart of the GBPUSD symbol has been opened with a period of M1 and ChartID 1
  */
 }
```

© 2000-2025, MetaQuotes Ltd.

ChartFirst

클라이언트 터미널의 첫 번째 차트의 ID를 반환

```
long  ChartFirst();
```

값 반환

차트 ID.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 클라이언트 터미널에 열린 첫 번째 차트의 ID를 가져옵니다.
   long            chart_id= ChartFirst();

//--- 획득한 ID를 사용하여 차트 심볼 및 기간을 추가로 가져옵니다.
   string          symbol  = ChartSymbol(chart_id);
   ENUM_TIMEFRAMES period  = ChartPeriod(chart_id);

//--- 클라이언트 터미널의 첫 번째 차트 설명을 저널에 표시합니다.
   PrintFormat("ID of the first chart of the client terminal: %I64d, chart symbol: %s,
   /*
   결과:
   ID of the first chart of the client terminal: 133246248352168440, chart symbol: EUR
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartNext

지정한 차트 옆에 있는 차트의 차트 ID를 반환.

```
long  ChartNext(
   long  chart_id      // 차트 ID
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미하지 않습니다. 0 은 "첫 번째 차트 ID 반환"을 의미합니다.

값 반환

차트 ID. 만약 이것이 차트 목록의 마지막이면 -1을 반환합니다.

예를 들어:

```
//--- 차트 ID의 변수
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst =",ChartSymbol(prevChart)," ID =",prevChart);
   while(i<limit)// 오픈 차트가 100개를 넘지 않는 것은 확실 합니다
     {
      currChart=ChartNext(prevChart); // 이전 차트 ID를 사용하여 새 차트 ID를 가져오기
      if(currChart<0) break;          // 차트 목록의 끝에 도달
      Print(i,ChartSymbol(currChart)," ID =",currChart);
      prevChart=currChart;// ChartNext()의 현재 차트 ID를 저장하겠습니다
      i++;// 카운터를 늘리는 것을 잊지 마십시오
     }
```

© 2000-2025, MetaQuotes Ltd.

ChartClose

지정된 차트를 닫기.

```
bool  ChartClose(
   long  chart_id=0      // 차트 ID
   );
```

매개변수

chart_id=0

[in] 차트 ID. 0은 현재 차트를 의미합니다.

값 반환

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트와 동일한 심볼 및 기간을 사용하여 새 차트를 엽니다. one
   long chart_id=ChartOpen(_Symbol, _Period);
   if(chart_id==0)
     {
      Print("ChartOpen() failed. Error ", GetLastError());
      return;
     }

//--- 저널에서 오픈 차트 매개변수를 인쇄합니다.
   PrintFormat("A new chart of the %s symbol has been opened with a period of %s and C
               _Symbol, StringSubstr(EnumToString(_Period), 7), chart_id);

//--- 2초를 기다리고 이후 새로 열린 차트를 닫습니다
   PrintFormat("Waiting 3 seconds before closing a newly opened chart with ID %I64d ..
   Sleep(3000);
   ResetLastError();
   if(!ChartClose(chart_id))
     {
      Print("ChartClose() failed. Error ", GetLastError());
      return;
     }

//--- 저널에 성공적인 차트 종료 메시지를 인쇄합니다.
   PrintFormat("The chart with ID %I64d was successfully closed", chart_id);
   /*
   결과:
   A new chart of the GBPUSD symbol has been opened with a period of M1 and ChartID 13
```

© 2000-2025, MetaQuotes Ltd.

```
   Waiting 3 seconds before closing a newly opened chart with ID 133346697706632016 ..
   The chart with ID 133346697706632016 was successfully closed
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSymbol

지정된 차트의 심볼 이름을 반환

```
string  ChartSymbol(
   long  chart_id=0      // 차트 ID
   );
```

매개변수

chart_id=0

[in] 차트 ID. 0은 현재 차트를 의미합니다.

값 반환

만약 차트가 없는 경우 결과는 빈 문자열이 됩니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 심볼을 가져오고 얻은 값을 저널에 표시합니다.
   string chart_symbol = ChartSymbol();
   Print("Current chart symbol: ", chart_symbol);

//--- 현재 차트 ID를 가져옵니다
   long chart_id=ChartID();
   chart_symbol=ChartSymbol(chart_id);
   PrintFormat("Chart symbol with ID %I64d: %s", chart_id, chart_symbol);

//--- 심볼을 수신할 때 임의의 차트 ID를 설정합니다.
   chart_symbol = ChartSymbol(1234567890);
   if(chart_symbol=="")
      Print("The chart with ID 1234567890 does not exist");
   else
      Print("Chart symbol with ID 1234567890: ", chart_symbol);
   /*
   결과:
   Current chart symbol: GBPUSD
   Chart symbol with ID 132966427583395104: GBPUSD
   The chart with ID 1234567890 does not exist
   */
  }
```

추가 참조

ChartSetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

ChartPeriod

지정된 차트의 시간프레임 기간을 반환.

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // 차트 ID
   );
```

매개변수

chart_id=0

[in] 차트 ID. 0은 현재 차트를 의미합니다.

값 반환

이 기능은 ENUM_TIMEFRAMES 값 중 하나를 반환합니다. 차트가 없으면 0을 반환합니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 주기를 가져오고 얻은 값을 저널에 표시
   ENUM_TIMEFRAMES period=ChartPeriod();
   Print("Current chart period: ", EnumToString(period));

//--- 현재 차트 ID를 가져옵니다
   long chart_id=ChartID();
   period=ChartPeriod(chart_id);
   PrintFormat("Chart period with ID %I64d: %s", chart_id, EnumToString(period));

//--- 랜덤 차트 ID를 설정
   period=ChartPeriod(1234567890);
   if(period==0)
      Print("The chart with ID 1234567890 does not exist");
   else
      Print("Chart period with ID 1234567890: ", EnumToString(period));
   /*
   결과:
   Current chart period: PERIOD_M15
   Chart period with ID 133510090240498505: PERIOD_M15
   The chart with ID 1234567890 does not exist
   */
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartRedraw

이 기능은 지정된 차트의 강제 재 작성을 호출.

```
void  ChartRedraw(
   long  chart_id=0      // 차트 ID
   );
```

매개변수

chart_id=0

[in] 차트 ID. 0은 현재 차트를 의미합니다.

주의

일반적으로 개체 속성을 변경한 후 사용됩니다.

예:

```
#define   WIDTH      50    // 막대 단위의 직사각형 너비

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 ID를 가져오고 그래픽 객체 이름을 설정합니다.
   long   chart_id = ChartID();
   string obj_name = MQLInfoString(MQL_PROGRAM_NAME)+"_RectLabel";

//--- 두 개의 직사각형 시간 좌표를 얻습니다.
   datetime time1  = iTime(_Symbol, _Period, WIDTH);
   datetime time2  = iTime(_Symbol, _Period, 0);
   if(time1==0 || time2==0)
     {
      Print("Error getting time ", GetLastError());
      return;
     }

//--- 직사각형 너비 범위에서 최대 및 최소 가격을 가져옵니다.
   double price1 = HighestHigh(_Symbol, _Period, 0, WIDTH);
   double price2 = LowestLow(_Symbol, _Period, 0, WIDTH);
   if(price1==EMPTY_VALUE || price2==EMPTY_VALUE)
      return;

//--- 직사각형 객체 생성
   Print("Create a wheat-colored rectangle");
   if(!ObjectCreate(chart_id, obj_name, OBJ_RECTANGLE, 0, time1, price1, time2, price2
     {
      Print("ObjectCreate() failed. Error ", GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- 직사각형을 원래의 색으로 채웁니다
   ObjectSetInteger(chart_id, obj_name, OBJPROP_FILL, true);
   ObjectSetInteger(chart_id, obj_name, OBJPROP_BACK, true);
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrWheat);
   ChartRedraw();

//--- 잠시 기다려서 직사각형을 DodgerBlue 색상으로 채우고 차트를 업데이트합니다
   Sleep(1000);
   Print("Change color to DodgerBlue");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrDodgerBlue);
   ChartRedraw();

//--- 잠시 기다려서 직사각형을 LimeGreen 색상으로 채우고 차트를 업데이트합니다
   Sleep(1000);
   Print("Change color to LimeGreen");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrLimeGreen);
   ChartRedraw();

//--- 잠시 기다려서 직사각형을 OrangeRed 색상으로 채우고 차트를 업데이트합니다
   Sleep(1000);
   Print("Change color to OrangeRed");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrOrangeRed);
   ChartRedraw();

//--- 잠시 기다려서 직사각형을 밀 색상으로 채우고 차트를 업데이트합니다
   Sleep(1000);
   Print("Reset color to original");
   ObjectSetInteger(chart_id, obj_name, OBJPROP_COLOR, clrWheat);
   ChartRedraw();

//--- 3초 후 차트에서 객체를 제거합니다.
   Sleep(3000);
   Print("Delete the rectangle");
   ObjectDelete(chart_id, obj_name);
  }
//+------------------------------------------------------------------+
//| 지정된 바의 범위에서 최대 High를 반환합니다.                          |
//+------------------------------------------------------------------+
double HighestHigh(const string symbol, const ENUM_TIMEFRAMES timeframe, const uint st
  {
   ResetLastError();
   int index=iHighest(symbol, timeframe, MODE_HIGH, count, start);
   if(index==-1)
     {
      PrintFormat("%s: iHighest() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   GetLastError();
   double price=iHigh(symbol, timeframe, index);
   if(price==0)
     {
      PrintFormat("%s: iHigh() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   return(price);
  }
//+------------------------------------------------------------------+
//| 지정된 바의 범위에서 최소 Low를 반환합니다.                      |
//+------------------------------------------------------------------+
double LowestLow(const string symbol, const ENUM_TIMEFRAMES timeframe, const uint sta
  {
   ResetLastError();
   int index=iLowest(symbol, timeframe, MODE_LOW, count, start);
   if(index==-1)
     {
      PrintFormat("%s: iLowest() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   GetLastError();
   double price=iLow(symbol, timeframe, index);
   if(price==0)
     {
      PrintFormat("%s: iLow() failed. Error %d",__FUNCTION__, GetLastError());
      return(EMPTY_VALUE);
     }
   return(price);
  }
```

추가 참조

개체 함수

© 2000-2025, MetaQuotes Ltd.

ChartSetDouble

지정한 차트의 해당 속성에 대한 값을 설정합니다. 차트 속성은 double 유형이어야 합니다. 이 명령은 차트 메시지 대기열에 추가되며 모든 이전 명령을 처리한 후 실행됩니다.

```
bool  ChartSetDouble(
   long                          chart_id,      // 차트 ID
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,       // 속성 ID
   double                        value          // 값
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

prop_id

[in] 차트 속성 ID. ENUM_CHART_PROPERTY_DOUBLE의 값 중 하나일 수 있습니다(읽기 전용 속성 제외).

값

[in] 속성 값.

값 반환

명령이 차트 대기열에 추가된 경우 true를 반환, 그렇지 않은 경우 false를 반환. 오류에 대한 정보를 가져오기 위해, GetLastError() 함수 호출.

주의

함수는 비동기식이며, 이는 함수가 지정된 차트의 대기열에 성공적으로 추가된 명령 실행을 기다리지 않음을 의미합니다. 대신, 컨트롤을 즉시 반환 차트 대기열에서 적절한 명령을 처리한 후에만 속성이 변경됩니다. 차트 대기열에서 명령을 즉시 시행하려면 ChartRedraw 함수를 호출.

한 번에 여러 차트 속성을 즉시 변경하려면 해당 함수(ChartSetString, ChartSetDouble, ChartSetString)를 하나의 코드 블록에서 실행해야 하며그런 다음 ChartRedraw를 한번 호출.

명령 실행 결과를 확인하려면 지정된 차트 속성 (ChartGetInteger, ChartGetDouble, ChartSetString)을 요청하는 함수를 사용할 수 있습니다. 하지만 이러한 기능은 동기화되어 실행 결과를 기다립니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 ID 가져오기
   long chart_id = ChartID();

//--- 오른쪽 가장자리에서 제로 바의 초기 오프셋을 백분율로 저장하고 플래그를 자동으로 스크롤
   double shift = ChartGetDouble(chart_id, CHART_SHIFT_SIZE);
   bool   scroll= ChartGetInteger(chart_id,CHART_AUTOSCROLL);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 자동 스크롤 차트 플래그를 재설정
   ChartSetInteger(chart_id, CHART_AUTOSCROLL, true);
   PrintFormat("Initial chart shift size: %.2f", shift);

//--- 루프에서 0.5% 간격으로 2.0%에서 50.0%까지
   for(int i=20; i<=500; i+=5)
     {
      //--- 이동 사이즈를 i/10으로 설정하고 저널에 지정된 값을 출력합니다
      ChartSetDouble(chart_id, CHART_SHIFT_SIZE, i/10.0);
      PrintFormat("Set chart shift size to %.1f%%", i/10.0);

      //--- 지정된 이동 거리에서 차트 제로 바 유지
      ChartNavigate(chart_id, CHART_END, 0);

      //--- 잠시 대기
      Sleep(16);
     }

//--- 초기 차트 이동 및 자동 스크롤 설정
   Print("Set the chart shift size to the initial value of ", shift);
   ChartSetDouble(ChartID(), CHART_SHIFT_SIZE, shift);
   ChartSetInteger(chart_id, CHART_AUTOSCROLL, scroll);
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSetInteger

지정한 차트의 해당 속성에 대한 값을 설정합니다. 차트 속성은 datetime, int, color, bool 또는 char여야 합니다. 이 명령은 차트 메시지 대기열에 추가되며 모든 이전 명령을 처리한 후 실행됩니다.

```
bool  ChartSetInteger(
   long                          chart_id,      // 차트 ID
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // 속성 ID
   long                          value          // 값
   );
```

지정된 하위 창의 해당 속성에 대한 값을 설정.

```
bool  ChartSetInteger(
   long                          chart_id,      // 차트 ID
   ENUM_CHART_PROPERTY_INTEGER   prop_id,       // 속성 ID
   int                           sub_window,    // 하위 창 번호
   long                          value          // 값
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

prop_id

[in] 차트 속성 ID. ENUM_CHART_PROPERTY_INTEGER 값 중 하나일 수 있습니다(읽기 전용 속성 제외).

sub_window

[in] 차트 하위창의 번호. 첫 번째 경우 기본 값은 0(주 차트장) 입니다. 대부분의 속성은 하위창 번호가 일치하지 않습니다.

값

[in] 속성 값.

값 반환

명령이 차트 대기열에 추가된 경우 true를 반환, 그렇지 않은 경우 false를 반환. 오류에 대한 정보를 가져오기 위해, GetLastError() 함수 호출.

주의

함수는 비동기식이며, 이는 함수가 지정된 차트의 대기열에 성공적으로 추가된 명령 실행을 기다리지 않음을 의미합니다. 대신, 컨트롤을 즉시 반환 차트 대기열에서 적절한 명령을 처리한 후에만 속성이 변경됩니다. 차트 대기열에서 명령을 즉시 시행하려면 ChartRedraw 함수를 호출.

한 번에 여러 차트 속성을 즉시 변경하려면 해당 함수(ChartSetString, ChartSetDouble, ChartSetString)를 하나의 코드 블록에서 실행해야 하며그런 다음 ChartRedraw를 한번 호출.

명령 실행 결과를 확인하려면 지정된 차트 속성 (ChartGetInteger, ChartGetDouble, ChartSetString)을 요청하는 함수를 사용할 수 있습니다. 하지만 이러한 기능은 동기화되어 실행 결과를 기다립니다.

예를 들어:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 함수                                               |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 차트 창에서 마우스 이동 이벤트 활성화
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- 차트 속성의 강제 업데이트로 이벤트 처리 준비 보장
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // 마우스 좌측
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // 마우스 우측
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // 마우스 중간
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // 마우스 첫 번째 X 키
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // 마우스 두 번째 X 키
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // 시프트 키
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // 컨트롤 키
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent 기능                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,const long &lparam,const double &dparam,const string &s
  {
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
  }
```

© 2000-2025, MetaQuotes Ltd.

ChartSetString

지정된 차트의 해당 속성에 대한 값을 설정. 차트 속성은 문자열 유형이어야 합니다. 이 명령은 차트 메시지 대기열에 추가되며 이전 모든 명령을 처리한 후 실행됩니다.

```
bool  ChartSetString(
   long                          chart_id,      // 차트 ID
   ENUM_CHART_PROPERTY_STRING    prop_id,       // 속성 ID
   string                        str_value      // 값
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

prop_id

[in] 차트 속성 ID. 그 값은 ENUM_CHART_PROPERTY_STRING 값 중 하나일 수 있습니다(읽기 전용 속성 제외).

str_value

[in] 속성 값 문자열. 문자열 길이는 2045자를 초과할 수 없습니다(초과 문자는 잘립니다).

값 반환

명령이 차트 대기열에 추가된 경우 true를 반환, 그렇지 않은 경우 false를 반환. 오류에 대한 정보를 가져오기 위해, GetLastError() 함수 호출.

주의

ChartSetString은 코멘트 기능 대신 차트의 코멘트 출력에 사용할 수 있습니다.

함수는 비동기식이며, 이는 함수가 지정된 차트의 대기열에 성공적으로 추가된 명령 실행을 기다리지 않음을 의미합니다. 대신, 컨트롤을 즉시 반환 차트 대기열에서 적절한 명령을 처리한 후에만 속성이 변경됩니다. 차트 대기열에서 명령을 즉시 시행하려면 ChartRedraw 함수를 호출.

한 번에 여러 차트 속성을 즉시 변경하려면 해당 함수(ChartSetString, ChartSetDouble, ChartSetString)를 하나의 코드 블록에서 실행해야 하며그런 다음 ChartRedraw를 한번 호출.

명령 실행 결과를 확인하려면 지정된 차트 속성 (ChartGetInteger, ChartGetDouble, ChartSetString)을 요청하는 함수를 사용할 수 있습니다. 하지만 이러한 기능은 동기화되어 실행 결과를 기다립니다.

예를 들어:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
   string comment=StringFormat("Выводим цены:\nAsk = %G\nBid = %G\nSpread = %d",
                               Ask,Bid,Spread);
```

© 2000-2025, MetaQuotes Ltd.

```
   ChartSetString(0,CHART_COMMENT,comment);
  }
```

추가 참조

Comment, ChartGetString

© 2000-2025, MetaQuotes Ltd.

ChartGetDouble

지정된 차트의 해당 속성 값을 반환. 차트 속성은 double 유형이어야 합니다. 함수 호출의 2가지 변수가 있습니다.

1. 속성 값을 직접 반환.

```
double  ChartGetDouble(
   long                        chart_id,         // 차트 ID
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // 속성 ID
   int                         sub_window=0      // 필요에 따라 하위창 번호
   );
```

2. 함수의 성공 여부에 따라 true 또는 false를 반환 성공하면 속성값이 참조로 전달된 대상 변수 double_var에 배치됩니다.

```
bool  ChartGetDouble(
   long                        chart_id,         // 차트 ID
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // 속성 ID
   int                         sub_window,       // 하위창 번호
   double&                     double_var        // 차트 속성의 대상 변수
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

prop_id

[in] 차트 속성 ID. 이 값은 ENUM_CHART_PROPERTY_DOUBLE 값 중 하나일 수 있습니다.

sub_window

[in] 차트 하위창의 번호. 첫 번째 경우 기본 값은 0(주 차트장) 입니다. 대부분의 속성은 하위창 번호가 일치하지 않습니다.

double_var

[out] 요청한 속성의 double 유형의 대상 변수.

값 반환

double 유형 값.

두 번째 호출 케이스의 경우, 지정된 속성을 사용할 수 있고 해당 값이 double_var 변수에 배치된 경우 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 추가 정보를 얻으려면 GetLastError()기능을 호출해야 합니다.

주의

이 기능은 동기식이며, 이는 호출하기 전에 차트 대기열에 추가된 모든 명령의 실행을 기다린다는 의미입니다.

예를 들어:

```
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
{
 double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
 double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
 Print("CHART_PRICE_MIN =",priceMin);
 Print("CHART_PRICE_MAX =",priceMax);
}
```

© 2000-2025, MetaQuotes Ltd.

ChartGetInteger

지정한 차트의 해당 속성 값을 반환. 차트 속성은 datetime, int 또는 bool 유형이어야 합니다. 함수 호출에는 두 가지 변수가 있습니다.

1. 속성 값을 직접 반환.

```
long  ChartGetInteger(
   long                          chart_id,         // 차트 ID
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // 속성 ID
   int                           sub_window=0      // 필요에 따라 하위창 번호
   );
```

2. 함수의 성공 여부에 따라 true 또는 false를 반환 성공하면 속성 값이 참조로 전달된 대상 변수 long_var에 배치됩니다.

```
bool  ChartGetInteger(
   long                          chart_id,         // 차트 ID
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // 속성 ID
   int                           sub_window,       // 하위 창 번호
   long&                         long_var          // 속성의 대상 변수
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

prop_id

[in] 차트 속성 ID. 값은 ENUM_CHART_PROPERTY_INTEGER 값 중 하나일 수 있습니다.

sub_window

[in] 차트 하위창의 번호. 첫 번째 경우 기본 값은 0(주 차트장) 입니다. 대부분의 속성은 하위창 번호가 일치하지 않습니다.

long_var

[out] 요청한 속성의 long 유형의 대상 변수.

값 반환

long 유형의 값.

두 번째 호출 케이스의 경우, 지정된 속성을 사용할 수 있고 해당 값이 long_var 변수에 저장된 경우 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 추가 정보를 얻으려면 GetLastError()기능을 호출해야 합니다.

주의

이 기능은 동기식이며, 이는 호출하기 전에 차트 대기열에 추가된 모든 명령의 실행을 기다린다는 의미입니다.

예를 들어:

```
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
{
 int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
 int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
 Print("CHART_HEIGHT_IN_PIXELS =",height,"pixels");
 Print("CHART_WIDTH_IN_PIXELS =",width,"pixels");
}
```

© 2000-2025, MetaQuotes Ltd.

ChartGetString

지정된 차트의 해당 속성 값을 반환. 차트 속성은 string 유형이어야 합니다. 함수 호출의 2가지 변수가 있습니다.

1. 속성 값을 직접 반환.

```
string  ChartGetString(
   long                          chart_id,          // 차트 ID
   ENUM_CHART_PROPERTY_STRING    prop_id            // 속성 ID
   );
```

2. 함수의 성공 여부에 따라 true 또는 false를 반환 성공하면 속성값이 참조로 전달된 대상 변수 string_var에 배치됩니다.

```
bool  ChartGetString(
   long                          chart_id,          // 차트 ID
   ENUM_CHART_PROPERTY_STRING    prop_id            // 속성 ID
   string&                       string_var         // 속성의 대상 변수
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

prop_id

[in] 차트 속성 ID. 이 값은 ENUM_CHART_PROPERTY_STRING 값 중 하나일 수 있습니다.

string_var

[out] 요청한 속성의 string 유형의 대상 변수.

값 반환

string 유형 값.

두 번째 호출 케이스의 경우, 지정된 속성을 사용할 수 있고 해당 값이 string_var 변수에 저장된 경우 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 추가 정보를 얻으려면 GetLastError()기능을 호출해야 합니다.

주의

ChartGetString는 Comment 또는 ChartSetString 기능을 사용하여 차트에 표시된 코멘트를 읽는 데 사용할 수 있습니다.

이 기능은 동기식이며, 이는 호출하기 전에 차트 대기열에 추가된 모든 명령의 실행을 기다린다는 의미입니다.

예를 들어:

```
void OnStart()
  {
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
   ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(1000);
   string comm=ChartGetString(0,CHART_COMMENT);
   Print(comm);
  }
```

추가 참조

Comment, ChartSetString

© 2000-2025, MetaQuotes Ltd.

ChartNavigate

차트에서 지정된 위치를 기준으로 지정된 막대 수만큼 지정된 차트의 이동을 수행.

```
bool  ChartNavigate(
   long                  chart_id,      // 차트 ID
   ENUM_CHART_POSITION   position,      // 포지션
   int                   shift=0        // 이동 값
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

포지션

[in] 이동을 수행할 차트 포지션 ENUM_CHART_POSITION 값 중 하나일 수 있습니다.

shift=0

[in] 차트를 이동할 막대 수. 양수 값은 오른쪽 이동(차트의 끝)을 의미하고 음수 값은 왼쪽(차트의 시작)을 의미합니다. 영점 이동을 사용하여 차트의 시작 또는 끝으로 이동할 수 있습니다.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트의 핸들 가져오기
   long handle=ChartID();
   string comm="";
   if(handle>0) // 성공적이면 차트를 추가로 설정합니다
     {
      //--- 자동 스크롤 비활성
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- 오른쪽 차트 경계에서 이동을 설정합니다.
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- 캔들스틱 작성
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- 틱 볼륨의 표시 모드를 설정
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

      //--- Comment()에 출력할 텍스트를 준비
      comm="내역 시작의 오른쪽으로 10개의 막대를 스크롤";
      //--- 코멘트 보기
      Comment(comm);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 내역 시작의 오른쪽으로 10개의 막대를 스크롤
      ChartNavigate(handle,CHART_BEGIN,10);
      //--- 차트에 표시되는 첫 번째 막대의 수를 가져오기(시계열과 같은 숫자)
      long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      //--- 라인 피드 문자 추가
      comm=comm+"\r\n";
      //--- 코멘트에 추가
      comm=comm+"차트의 첫 번째 막대는 숫자입니다 "+IntegerToString(first_bar)+"\r\n";
      //--- 코멘트 보기
      Comment(comm);
      //--- 5초 동안 차트의 이동 방법을 확인
      Sleep(5000);

      //--- 코멘트 텍스트에 추가
      comm=comm+"\r\n"+"10개의 막대를 오른쪽 차트 데두리의 왼쪽으로 스크롤";
      Comment(comm);
      //--- 10개의 막대를 오른쪽 차트 테두리 왼쪽으로 스크롤
      ChartNavigate(handle,CHART_END,-10);
      //--- 차트에 표시되는 첫 번째 막대의 수를 가져오기(시계열과 같은 숫자)
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
      comm=comm+"차트의 첫 번째 막대는 숫자입니다 "+IntegerToString(first_bar)+"\r\n";
      Comment(comm);
      //--- 5초 동안 차트의 이동 방법을 확인
      Sleep(5000);

      //--- 차트 스크롤의 새 블록
      comm=comm+"\r\n"+"300개의 막대를 내역 시작의 오른쪽으로 스크롤";
      Comment(comm);
      //--- 300개의 막대를 내역 시작의 오른쪽으로 스크롤
      ChartNavigate(handle,CHART_BEGIN,300);
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
      comm=comm+"차트의 첫 번째 막대는 숫자입니다 "+IntegerToString(first_bar)+"\r\n";
      Comment(comm);
      //--- 5초 동안 차트의 이동 방법을 확인
      Sleep(5000);

      //--- 차트 스크롤의 새 블록
      comm=comm+"\r\n"+"300개의 막대를 오른쪽 차트의 왼쪽으로 스크롤";
      Comment(comm);
      //--- 300개의 막대를 오른쪽 차트 테두리 왼쪽으로 스크롤
      ChartNavigate(handle,CHART_END,-300);
      first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
      comm=comm+"\r\n";
      comm=comm+"차트의 첫 번째 막대는 숫자입니다 "+IntegerToString(first_bar)+"\r\n";
      Comment(comm);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

ChartID

현재 차트의 ID를 입력하십시오.

```
long  ChartID();
```

반환값

롱 타입의 값.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 차트 식별자 변수들
   long curr_chart=ChartFirst();
   int i=0;
   //--- 첫 번째 차트 데이터를 저널에 출력
   PrintFormat("Chart[%d] ID: %I64d,  symbol: %s", i, curr_chart, ChartSymbol(curr_cha

//--- 오픈 차트의 최대 한도에 도달할 때까지(CHARTS_MAX)
   while(!IsStopped() && i < CHARTS_MAX)
     {
      //--- 차트 카운터 증가
      i++;
      //--- 이전 차트를 기반으로 다음 차트 ID 가져오기
      curr_chart=ChartNext(curr_chart);

      //--- 차트 목록 끝에 도달하면 루프를 종료합니다
      if(curr_chart<0)
         break;

      //--- 다음 차트 데이터를 저널에 출력
      PrintFormat("Chart[%d] ID: %I64d,  symbol: %s", i, curr_chart, ChartSymbol(curr_
     }
   /*
   결과:
   Chart[0] ID: 133246248352168440,  symbol: EURUSD
   Chart[1] ID: 133346697706632015,  symbol: USDJPY
   Chart[2] ID: 133246248352168439,  symbol: GBPUSD
   Chart[3] ID: 133346697706632009,  symbol: RU000A103661
   Chart[4] ID: 133346697706632010,  symbol: AEM4
   Chart[5] ID: 133346697706632011,  symbol: AA.SPB
   Chart[6] ID: 133346697706632012,  symbol: ALLFUTMIX
   Chart[7] ID: 133346697706632013,  symbol: EURUSD
   Chart[8] ID: 133346697706632014,  symbol: SBER
   */
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorAdd

지정된 핸들이 있는 지표를 지정된 차트 창에 추가. 지표 및 차트는 동일한 심볼과 시간대에 생성되어야 합니다.

```
bool  ChartIndicatorAdd(
   long  chart_id,                  // 차트 ID
   int   sub_window                 // 하위창 번호
   int   indicator_handle           // 지표의 핸들
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

sub_window

[in] 차트 하위창의 번호 0은 주 차트 창을 의미합니다. 새 창에 지표를 추가하려면 변수가 마지막 기존 창의 인덱스, 즉 CHART_WINDOWS_TOTAL와 동일한 만큼 커야 합니다. 변수의 값이 CHART_WINDOWS_TOTAL보다 크면 새 창이 생성되지 않고 지표가 추가되지 않습니다.

indicator_handle

[in] 지표의 핸들.

값 반환

이 함수는 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻기 위해서는, GetLastError() 기능을 호출. 오류 4114는 차트와 추가된 지표가 기호나 시간대에 따라 다르다는 것을 의미합니다.

주의

별도의 하위 창에 작성되어야 하는 지표 (예를 들어, 내장형 iMACD 또는 지정된#property indicator_separate_window 속성을 가진 지표)가 기본 차트 창에 적용되면 지표 목록에 계속 존재하더라도 표시되지 않을 수 있습니다. 즉, 지표의 척도가 가격 차트의 척도와 다르며 적용된 지표의 값이 가격 차트의 표시된 범위에 맞지 않습니다. 이런 경우, GetLastError()는 오류의 부재를 나타내는 0코드를 반환합니다. 이러한 "보이지 않는" 지표의 값은 데이터 윈도우에서 볼 수 있으며 다른 MQL5 애플리케이션에서 수신할 수 있습니다.

예를 들어:

```
#property 설명 "엑스퍼트 어드바이저는 ChartIndicatorAdd() 기능을 사용하여 작업을 시연합니다."
#property 설명 "차트에서 시작한 후 (저널에 오류를 수신하고), 열기"
#property 설명 "엑스퍼트 어드바이저의 속성을 확인하고 올바른 <symbol> 및 <period> 변수를 지정
#property 설명 "MACD 지표가 차트에 추가됩니다."

//--- 변수 입력
input string          symbol="AUDUSD";     // 심볼 이름
input ENUM_TIMEFRAMES period=PERIOD_M12;   // 시간 플레임
input int     fast_ema_period=12;          // 빠른 MACD 주기
input int     slow_ema_period=26;          // 느린 MACD 주기
input int       signal_period=9;           // 시그널 주기
```

© 2000-2025, MetaQuotes Ltd.

```
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // MACD의 가격 유형 calculation

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- 차트에 지표를 추가하려고 합니다
   if(!AddIndicator())
     {
      //--- AddIndicator() 기능이 차트에서 지표 추가를 거부했습니다
      int answer=MessageBox("차트에 MACD를 추가하시겠습니까?",
                            "지표를 추가하기 위한 잘못된 심볼 및 시간 프레임",
                            MB_YESNO // "예" 및 "아니오" 선택 버튼이 표시됩니다
                            );
      //--- 사용자가 여전히 ChartIndicatorAdd()의 잘못된 사용을 주장하는 경우
      if(answer==IDYES)
        {
         //--- 우선, 그것에 대한 저널에 등록할 것입니다
         PrintFormat("주의! %s: MACD(%s/%s) 지표를 %s/%s 차트에 추가를 시도. 오류 4114를 수
                     __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
         //--- 지표를 추가할 때 새 하위 창의 번호를 수신
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- 오류 발생 시도를 합니다
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("MACD 지표를 %d 차트 창에 추가하는 데 실패하였습니다. 오류 코드  %d"
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 함수                                                  |
//+------------------------------------------------------------------+
void OnTick()
  {
// 엑스퍼트 어드바이저가 아무것도 수행하지 않음
  }
//+------------------------------------------------------------------+
//| 차트에서 지표 확인 및 추가 함수                                    |
//+------------------------------------------------------------------+
bool AddIndicator()
  {
//--- 메시지 표시됨
   string message;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 지표 심볼과 차트 심볼이 서로 일치하는지 확인
   if(symbol!=_Symbol)
     {
      message="Demo_ChartIndicatorAdd() 기능 사용을 표시:";
      message=message+"\r\n";
      message=message+"차트의 다른 심볼에 계산된 지표를 추가할 수 없습니다.";
      message=message+"\r\n";
      message=message+"엑스퍼트 어드바이저의 속성에서 차트 심볼을 지정 - "+_Symbol+".";
      Alert(message);
      //--- 조기 종료, 지표가 차트에 추가되지 않음
      return false;
     }
//--- 지표와 차트의 시간 프레임이 서로 일치하는지 확인
   if(period!=_Period)
     {
      message="차트의 다른 시간 프레임에 계산된 지표를 추가할 수 없습니다.";
      message=message+"\r\n";
      message=message+"엑스퍼트 어드바이저의 속성에서 차트 시간 프레임을 지정 - "+EnumToStri
      Alert(message);
      //--- 조기 종료, 지표가 차트에 추가되지 않음
      return false;
     }
//--- 모든 점검 완료됨, 심볼 및 지표 시간 프레임이 차트에 일치
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__,"  MACD 지표 생성");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_peri
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("MACD 지표 생성하는 데 실패하였습니다. 오류 코드 ",GetLastError());
        }
     }
//--- 오류 코드 재설정
   ResetLastError();
//--- 차트에 지표 적용
   Print(__FUNCTION__,"  차트에 MACD 지표 추가");
   Print("MACD 에 다음에 생성됨 ",symbol,"/",EnumToString(period));
//--- MACD 지표가 추가되는 새 하위 창의 번호를 수신
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   PrintFormat("MACD 지표를 %d 차트 창에 추가",subwindow);
   if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
     {
      PrintFormat("MACD 지표를 %d 차트 창에 추가하는 데 실패하였습니다. 오류 코드  %d",
                  subwindow,GetLastError());
     }
//--- 지표가 성공적으로 추가됨
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

추가 참조

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

© 2000–2025, MetaQuotes Ltd.

ChartIndicatorDelete

지정된 차트 창에서 지정된 이름의 지표를 제거.

```
bool  ChartIndicatorDelete(
   long           chart_id,              // 차트 id
   int            sub_window             // 하위 창 번호
   const string   indicator_shortname    // 지표의 단축명
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 나타냅니다.

sub_window

[in] 차트 하위창의 번호. 0은 주 차트 하위 창을 나타냅니다.

const indicator_shortname

[in] INDICATOR_SHORTNAME 속성에 IndicatorSetString()기능에서 지표의 단축명을 설정. 지표의 단축명을 가져오기 위해 ChartIndicatorName() 기능을 사용.

값 반환

지표를 성공적으로 삭제할 경우 true를 반환합니다. 그렇지 않으면 false를 반환합니다. 오류 정보를 가져오기 위해 GetLastError() 기능을 사용.

주의

차트 하위 창에 같은 단축명을 가진 두 개의 지표가 있는 경우, 행의 첫 번째 지표가 삭제됩니다.

이 차트의 다른 지표가 삭제 중인 지표 값을 기반으로 하는 경우, 그러한 지표 또한 삭제 됩니다.

iCustom() 및 IndicatorCreate()기능을 사용하여 지표를 만들 때, 지정된 지표의 단축명과 파일 이름을 혼동하지 마십시오. 지표의 단축명이 명시적으로 설정되지 않는 경우, 지표의 소스코드를 포함하는 파일 이름이 컴파일 중에 지정됩니다.

차트에서 지표를 삭제한다고 해서 해당 계산 부분이 터미널 메모리에서 삭제되는 것은 아닙니다. 지표 핸들을 해제하기 위해 IndicatorRelease() 기능을 사용.

지표의 단축명이 올바른 형식이어야 합니다. INDICATOR_SHORTNAME 속성에 IndicatorSetString() 기능을 사용하여 기록됩니다. 단축명은 지표의 모든 입력 변수의 값을 포함해야하는데, 이는 ChartIndicatorDelete() 기능에 의해 차트에서 삭제될 지표의 단축명으로 식별되기 때문입니다.

초기화에 실패한 후 지표를 삭제하는 예:

```
//+------------------------------------------------------------------+
//|                                  Demo_ChartIndicatorDelete.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- 히스토그램 플롯
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- 지표 버퍼
double         HistogramBuffer[];
string         shortname;
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//--- HistogramBuffer 배열을 지표 버퍼에 연결
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- 입력 매개변수를 기준으로 지표의 단축명 구성
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- 지표의 강제 완료가 설정된 경우 0이 아닌 값을 반환합니다
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 루프에서 작업하기 위한 시작 포지션
   int start=prev_calculated-1;
```

© 2000-2025, MetaQuotes Ltd.

```
   if(start<0) start=0;
//--- 지표 버퍼에 값을 채우기
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=close[i];
     }
//--- 다음 번 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Deinit 이벤트의 핸들러                                           |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: 초기화 취소 이유 code=%d",__FUNCTION__,reason);
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("%s (파일 %s) 단축명을 가진 지표가 차트에서 자체 삭제됩니다",shortname,__F
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- ChartIndicatorDelete()의 호출 결과 분석
      if(!res)
        {
         PrintFormat("%s 지표를 #%d 창에서 삭제하는 데 실패하였습니다. 오류 코드 %d",
                     shortname,window,GetLastError());
        }
     }
  }
```

추가 참조

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorGet

지정된 차트 창에 지정된 단축명을 가진 지표의 핸들을 반환.

```
int  ChartIndicatorGet(
   long           chart_id,              // 차트 ID
   int            sub_window             // 하위 창의 번호
   const string   indicator_shortname    // 지표의 단축명
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

sub_window

[in] 차트 하위 창의 번호. 0은 주 차트 창을 의미합니다.

const indicator_shortname

[in] INDICATOR_SHORTNAME 속성에 IndicatorSetString() 기능을 사용하여 설정된 지표 경우의 단축명을 설정. 지표의 단축명을 가져오려면, ChartIndicatorName() 기능을 사용.

값 반환

성공한 경우 지표 핸들을 반환하고, 그렇지 않은 경우 INVALID_HANDLE를 반환합니다. 오류에 대한 정보를 얻으려면, GetLastError() 함수 호출.

주의

ChartIndicatorGet() 기능을 사용하여 얻은 지표 핸들은 내부 지표 사용 카운터를 증가시킵니다. 터미널 런타임 시스템은 카운터가 0보다 큰 모든 지표를 로드 상태로 유지합니다. 따라서 더 이상 필요하지 않은 지표 핸들은 아래 예와 같이 IndicatorRelease() 를 사용하여 즉시 그리고 명시적으로 해제해야 합니다. 그렇지 않다면 "abandoned" 핸들을 찾아서 다른 프로그램에서 올바르게 해제하는 것은 불가능할 겁니다.

지표를 생성할 때, INDICATOR_SHORTNAME 속성에 IndicatorSetString() 기능을 사용하여 기록되는 단축명을 주의하여 작성하십시오. 지표는 단축명을 기반으로 ChartIndicatorGet() 기능에서 식별되므로 단축명은 지표의 입력 매개변수 값을 포함하는 것이 좋습니다.

지표를 식별하는 또 다른 방법은 IndicatorParameters() 기능을 사용하여 주어진 핸들에 대한 매개변수 목록을 가져온 다음 얻은 값을 분석하는 것입니다.

예를 들어:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- 차트에 있는 창의 수(하나 이상의 기본 창이 항상 존재)
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   //--- 모든 창 확인
   for(int w=0;w<windows;w++)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- 이 창/하위 창의 지표 수
      int total=ChartIndicatorsTotal(0,w);
      //--- 창의 모든 지표 살펴보기
      for(int i=0;i<total;i++)
        {
         //--- 지표의 단축명 가져오기
         string name=ChartIndicatorName(0,w,i);
         //--- 지표 핸들 가져오기
         int handle=ChartIndicatorGet(0,w,name);
         //--- 로그에 추가
         PrintFormat("Window=%d,  index=%d,  name=%s,  handle=%d",w,i,name,handle);
         //--- 지표 핸들이 더 이상 필요하지 않을 때에는 반드시 해제해야 합니다.
         IndicatorRelease(handle);
        }
     }
  }
```

추가 참조

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), IndicatorParameters()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorName

지정된 차트 창의 지표 목록에 있는 숫자를 기준으로 지표의 단축명을 반환.

```
string  ChartIndicatorName(
   long  chart_id,       // 차트 id
   int   sub_window      // 하위 창의 번호
   int   index           // 차트 하위 창에 추가된 지표 목록의 지표 인덱스
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 나타냅니다.

sub_window

[in] 차트 하위창의 번호. 0은 주 차트 하위 창을 나타냅니다.

인덱스

[in] 지표의 목록에 있는 지표의 인덱스. 지표 수는 0으로 시작하는데, 즉 목록의 첫 번째 지표는 0 인덱스를 가집니다. 목록의 지표의 수를 얻기 위해 ChartIndicatorsTotal() 기능을 사용.

값 반환

INDICATOR_SHORTNAME 속성에 IndicatorSetString() 기능을 사용하여 설정된 지표의 단축명. 오류 정보를 가져오기 위해GetLastError() 기능을 사용.

주의

iCustom() 및 IndicatorCreate()기능을 사용하여 지표를 만들 때, 지정된 지표의 단축명과 파일 이름을 혼동하지 마십시오. 지표의 단축명이 명시적으로 설정되지 않은 경우 지표의 소스 코드를 포함하는 파일 이름이 컴파일 중에 지정됩니다.

지표의 단축명이 올바른 형식이어야 합니다. INDICATOR_SHORTNAME 속성에 IndicatorSetString() 기능을 사용하여 기록됩니다. 단축명은 지표의 모든 입력 변수의 값을 포함해야하는데, 이는 ChartIndicatorDelete() 기능에 의해 차트에서 삭제될 지표의 단축명으로 식별되기 때문입니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 ID와 기본 창을 포함한 하위 창의 수를 가져옵니다.
   long chart_id = ChartID();
   int  wnd_total= (int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);

//--- 루프의 모든 현재 차트 창을 반복합니다.
   for(int w=0; w<wnd_total; w++)
     {
      //--- 루프 인덱스로 지정된 차트 창에 연결된 지표의 수를 가져옵니다.
      int ind_total=ChartIndicatorsTotal(chart_id, w);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 선택된 차트 창의 헤더를 출력합니다
      PrintFormat("Chart window %d indicators: ", w);

      //--- 루프에 들어가서 선택한 창에 연결된 모든 지표의 이름을 변수에 씁니다.
      string ind_names="";
      for(int i=0; i<ind_total; i++)
        {
         ind_names+="  "+ChartIndicatorName(chart_id, w, i)+(i<ind_total-1 ? "\n": "");
        }
      //--- 지정된 차트 창에 첨부된 모든 지표 이름의 목록을 저널에 출력합니다.
      Print(ind_names);
     }
   /*
   결과:
   Chart window 0 indicators:
     AMA(9,2,30)
     SAR(0.02,0.20)
     Fractals
   Chart window 1 indicators:
     RSI(14)
     AMA(9,2,30)
   Chart window 2 indicators:
     MFI(14)
   */
  }
```

추가 참조

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000-2025, MetaQuotes Ltd.

ChartIndicatorsTotal

지정된 차트 창에 적용된 모든 지표의 수를 반환

```
int  ChartIndicatorsTotal(
   long  chart_id,      // 차트 id
   int   sub_window     // 하위 창의 번호
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 나타냅니다.

sub_window

[in] 차트 하위창의 번호. 0은 주 차트 하위 창을 나타냅니다.

값 반환

지정한 차트 창에 있는 지표의 수. 오류 정보를 가져오기 위해 GetLastError() 기능을 사용.

주의

이 기능을 사용하면 차트에 연결된 모든 지표를 검색할 수 있습니다. 차트의 모든 창 수는 CHART_WINDOWS_TOTAL 속성에서 ChartGetInteger() 함수를 사용하여 얻을 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 ID와 기본 창을 포함한 하위 창의 수를 가져옵니다.
   long chart_id = ChartID();
   int  wnd_total= (int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);

//--- 루프의 모든 현재 차트 창을 반복합니다.
   for(int w=0; w<wnd_total; w++)
     {
      //--- 루프 인덱스로 지정된 차트 창에 연결된 지표의 수를 가져옵니다.
      int ind_total=ChartIndicatorsTotal(chart_id, w);

      //--- 지정된 차트 창에 연결된 지표 수를 가져옵니다.
      PrintFormat("Chart ID %I64d, subwindow %d. Attached indicators: %d", chart_id, w
     }
   /*
   결과:
   Chart ID 133246248352168439, subwindow 0. Attached indicators: 3
   Chart ID 133246248352168439, subwindow 1. Attached indicators: 2
   Chart ID 133246248352168439, subwindow 2. Attached indicators: 1
   */
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

추가 참조

ChartIndicatorAdd(), ChartIndicatorDelete(), iCustom(), IndicatorCreate(), IndicatorSetString()

© 2000–2025, MetaQuotes Ltd.

ChartWindowOnDropped

엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 하위 창의 숫자(인덱스)를 반환. 0은 주 차트 창을 의미합니다.

```
int  ChartWindowOnDropped();
```

값 반환

int 유형의 값.

예를 들어:

```
   int myWindow=ChartWindowOnDropped();
   int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("스크립트가 # 창에서 실행되고 있습니다"+myWindow+
         ". 차트의 전체 창 "+ChartSymbol()+":",windowsTotal);
```

추가 참조

ChartPriceOnDropped, ChartTimeOnDropped, ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartPriceOnDropped

엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트에 해당하는 가격 좌표를 반환.

```
double  ChartPriceOnDropped();
```

값 반환

double 유형의 값.

예를 들어:

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

추가 참조

ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartTimeOnDropped

엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 지점에 해당하는 시간 좌표를 반환.

```
datetime  ChartTimeOnDropped();
```

값 반환

시간날짜 유형의 값.

예를 들어:

```
   datetime t=ChartTimeOnDropped();
   Print("스크립트가 다음에서 삭제되었습니다."+t);
```

추가 참조

ChartXOnDropped, ChartYOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartXOnDropped

엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트의 X좌표를 반환.

```
int  ChartXOnDropped();
```

값 반환

X좌표 값.

주의

왼쪽에서 오른쪽으로 X축 방향.

예를 들어:

```
   int X=ChartXOnDropped();
   int Y=ChartYOnDropped();
   Print("(X,Y) = ("+X+","+Y+")");
```

추가 참조

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartYOnDropped

EA 또는 스크립트가 떨어진 차트 포인트의 Y 좌표를 반환.

```
int  ChartYOnDropped();
```

값 반환

Y 좌표 값.

주의

Y 축 방향 위에서 아래로.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 마우스를 사용하여 스크립트를 놓은 차트 포인트의 X 및 Y 좌표를 가져오고 출력합니다.
   int x=ChartXOnDropped();
   int y=ChartYOnDropped();
   PrintFormat("Script dropped to coordinates X = %d, Y = %d", x, y);
   /*
   결과:
   Script dropped to coordinates X = 429, Y = 114
   */
  }
```

추가 참조

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

© 2000-2025, MetaQuotes Ltd.

ChartSetSymbolPeriod

지정된 차트의 기호와 기간을 변경 이 함수는 비동기식이며, 즉 명령을 전송하고 실행이 완료될 때까지 기다리지 않습니다. 이 명령은 차트 메시지 대기열에 추가되며 이전 모든 명령을 처리한 후 실행됩니다.

```
bool  ChartSetSymbolPeriod(
   long             chart_id,      // 차트 ID
   string           symbol,        // 심볼 이름
   ENUM_TIMEFRAMES  period         // 기간
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

심볼

[in] 차트 심볼. NULL 값은 현재 차트 기호를 의미합니다 (엑스퍼트 어드바이저가 첨부됨)

기간

[in] 차트 기간 (시간프레임). ENUM_TIMEFRAMES 값 중 하나일 수 있습니다. 0은 현재 차트 기간을 의미합니다.

값 반환

명령이 차트 대기열에 추가된 경우 true를 반환, 그렇지 않은 경우 false를 반환. 오류에 대한 정보를 가져오기 위해, GetLastError() 함수 호출.

주의

심볼/주기 변경으로 인해 차트에 첨부된 엑스퍼트 어드바이저가 재초기화됩니다.

심볼과 시간프레임이 동일한 ChartSetSymbolPeriod의 호출을 사용하여 차트를 업데이트할 수 있습니다 (터미널의 새로고침 명령과 유사). 차트 업데이트는 차례로 차트 업데이트에 연결된 지표 재계산을 트리거합니다. 따라서 틱이 없는 경우(예: 주말)에도 차트에서 지표를 계산할 수 있습니다.

예:

```
#define   SYMBOL    "GBPUSD"
#define   PERIOD    PERIOD_H1

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 현재 차트 ID, 심볼 및 기간을 가져옵니다.
   long            chart_id= ChartID();
   string          symbol  = Symbol();
   ENUM_TIMEFRAMES period  = Period();

//--- 현재 차트 심볼 및 기간을 SYMBOL 및 PERIOD에 설정된 것으로 바꾸는 것에 대해 저널에 보고합
   PrintFormat("Change the %s symbol and the %s period of the chart %I64u to %s %s",
```

© 2000-2025, MetaQuotes Ltd.

```
                symbol, TimeframeDescrioption(period), chart_id, SYMBOL, TimeframeDesc

//--- 차트 심볼과 주기를 변경합니다
   ChartSetSymbolPeriod(chart_id, SYMBOL, PERIOD);
   /*
  결과:
   Change the EURUSD symbol and the M1 period of the chart 133246248352168440 to GBPU
   */
  }
//+------------------------------------------------------------------+
//| 차트 기간 설명 반환                                              |
//+------------------------------------------------------------------+
string TimeframeDescrioption(const ENUM_TIMEFRAMES timeframe)
  {
   return(StringSubstr(EnumToString(timeframe), 7));
  }
```

추가 참조

ChartSymbol, ChartPeriod

© 2000-2025, MetaQuotes Ltd.

ChartScreenShot

이 함수는 지정된 확장자에 따라 GIF, PNG 또는 BMP 형식의 현재 상태 차트의 스크린샷을 제공.

```
bool  ChartScreenShot(
   long             chart_id,                  // 차트 ID
   string           filename,                  // 심볼 이름
   int              width,                     // 너비
   int              height,                    // 높이
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // 정렬 유형
   );
```

매개변수

chart_id

[in] 차트 ID. 0은 현재 차트를 의미합니다.

파일명

[in] 스크린샷 파일 이름. 63자를 초과할 수 없습니다. 스크린샷 파일은 \Files 디렉토리에 저장됩니다.

너비

[in] 스크린샷 너비(픽셀).

높이

[in] 스크린샷 높이(픽셀).

align_mode=ALIGN_RIGHT

[in] 좁은 스크린샷의 출력 모드. ENUM_ALIGN_MODE 열거의 값. ALIGN_RIGHT은 오른쪽 여백(끝으로부터의 출력)에 맞춰 정렬하는 것을 의미합니다. ALIGN_LEFT 은 좌측 정렬을 의미합니다.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

주의

특정 위치에서 타츠를 캡쳐해야 하는 경우, 먼저 ChartNavigate() 기능을 사용하여 그래프를 배치해야 합니다. 스크린샷의 수평 크기가 차트 창보다 작은 경우 align_mode 설정에 따라 차트 창의 오른쪽 부분 또는 왼쪽 부분이 출력됩니다.

예를 들어:

```
#property 설명 "엑스퍼트 어드바이저는 현재 스크린샷을 생성하는 방법을 시연합니다"
#property 설명 "ChartScreenShot() 기능을 사용하여 차트화. 편의를 위해, 파일의 이름은"
#property 설명 "차트에 표시됨. 이미지의 높이와 너비는 매크로를 통해 정의됩니다."

#define        WIDTH  800     // ChartScreenShot()을 호출할 이미지 너비
#define        HEIGHT 600     // ChartScreenShot() 호출할 이미지 높이

//--- 매개변수 입력
input int      pictures=5;    // 시리즈의 이미지 수
int            mode=-1;       // -1 은 차트의 오른쪽 가장자리로, 1은 왼쪽으로 이동을 나타냅니
```

© 2000-2025, MetaQuotes Ltd.

```
int            bars_shift=300;// ChartNavigate()를 사용하여 차트를 스크롤할 때의 막대 수
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 함수                                               |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 차트 자동 스크롤 사용 안 함
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- 차트의 오른쪽 가장자리의 이동을 설정
   ChartSetInteger(0,CHART_SHIFT,true);
//--- 캔들스틱 차트 표시
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("엑스퍼트 어드바이저의 준비 완료");
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 함수                                                  |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| ChartEvent 기능                                                  |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- 함수 이름, 호출 시간 및 이벤트 식별자 표시
   Print(__FUNCTION__,TimeCurrent(),"   id=",id,"   mode=",mode);
//--- CHARTEVENT_CLICK 이벤트 처리 ("차트에서 마우스를 클릭")
   if(id==CHARTEVENT_CLICK)
     {
      //--- 차트 가장자리에서 초기 이동
      int pos=0;
      //--- 왼쪽 차트 가장자리를 사용한 작업
      if(mode>0)
        {
         //--- 차트를 왼쪽 가장자리로 스크롤
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- 차트에 표시할 텍스트와 파일 이름 준비
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- 차트에 코멘트로 이름 표시
            Comment(name);
```

© 2000-2025, MetaQuotes Ltd.

```
            //--- 차트 스크린샷 파일을 terminal_directory\MQL5\Files\에 저장
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("스크린샷을 저장했습니다 ",name);
            //---
            pos+=bars_shift;
            //--- 사용자에게 차트의 새 부분을 살펴볼 시간을 주십시오
            Sleep(3000);
            //--- 현재 포지션 bars_shift 막대에서 오른쪽으로 스크롤
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- 모드를 반대로 변경
         mode*=-1;
        }
      else // 오른쪽 차트 가장자리를 사용한 작업
        {
         //--- 차트를 오른쪽 가장자리로 스크롤
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- 차트에 표시할 텍스트와 파일 이름 준비
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- 차트에 코멘트로 이름 표시
            Comment(name);
            //--- 차트 스크린샷 파일을 terminal_directory\MQL5\Files\에 저장
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("스크린샷을 저장했습니다 ",name);
            //---
            pos+=bars_shift;
            //--- 사용자에게 차트의 새 부분을 살펴볼 시간을 주십시오
            Sleep(3000);
            //--- 현재 포지션 bars_shift 막대에서 오른쪽으로 스크롤
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- 모드를 반대로 변경
         mode*=-1;
        }
     } // CHARTEVENT_CLICK 이벤트 핸들링 종료
//--- OnChartEvent() 처리 종료
  }
```

추가 참조

ChartNavigate(), Resources

© 2000-2025, MetaQuotes Ltd.

거래 함수

이는 거래 활동을 관리하기 위한 함수 그룹입니다.

플랫폼의 거래 함수를 학습하기 전에 주문, 거래 및 포지셔닝과 같은 기본 용어를 명확히 이해해야 합니다.

- 주문은 중개인에게 금융상품을 사거나 팔라고 지시하는 것입니다. 주문 유형에는 크게 마켓과 보류라는 두 가지가 있습니다. 또한, 특별 이익 실현 및 스탑 로스 레벨이 있습니다.
- 딜은 금융증권의 상업적 거래(매입 또는 매수)입니다. 매입은 요구가격(Ask)으로, 매도는 공급가격(Byd)으로 이루어집니다. 시장 주문 실행 또는 보류 중인 주문 트리거의 결과로 거래가 개시될 수 있습니다. 경우에 따라 주문이 실행되어 여러 거래가 발생할 수 있습니다.
- 포지션은 거래 의무, 즉 금융상품의 매입 또는 매도 계약의 수입니다. 롱 포지션은 유가 상승을 기대하며 사들인 금융 담보입니다. 숏 포지션은 미래에 가격이 하락할 것을 예상하여 담보를 공급해야 하는 의무입니다.

거래 작업의 일반적 정보는 다음에서 확인할 수 있습니다: 클라이언트 터미널 도움말.

Expert Advisor 및 스크립트에서 거래 함수를 사용할 수 있습니다. Expert Advisor 또는 스크립트에 있는 "실시간 거래 허용" 확인란이 활성화된 경우에만 거래 함수를 호출할 수 있습니다.

거래 허가 섹션에 설명된 다양한 요소에 따라 거래가 허용되거나 금지될 수 있습니다.

함수	기능
OrderCalcMargin	지정된 주문 유형에 필요한 마진을 예금 통화로 계산합니다
OrderCalcProfit	전달된 매개 변수를 기준으로 예치된 통화로 이익을 계산합니다
OrderCheck	필요한 거래 작업을 실행할 자금이 충분한지 확인합니다.
OrderSend	서버에 거래 요청을 보냅니다
OrderSendAsync	거래 서버의 거래 응답을 기다리지 않고 비동기식으로 거래 요청 전송
PositionsTotal	오픈 포지션 수를 반환합니다.
PositionGetSymbol	오픈 포지션에 해당하는 심볼을 반환합니다.
PositionSelect	추가 작업을 위해 오픈 포지션을 선택합니다
PositionSelectByTicket	지정된 티켓 번호로 작업할 포지션을 선택합니다
PositionGetDouble	오픈 포지션의 요청된 속성을 반환합니다 (double).
PositionGetInteger	오픈 포지션의 요청된 속성을 반환합니다 (datetime 또는 int)
PositionGetString	오픈 포지션의 요청된 속성을 반환합니다 (string)
PositionGetTicket	오픈 포지션 목록에 지정된 인덱스를 가진 포지션의 티켓을 반환합니다
OrdersTotal	주문 수를 반환합니다
OrderGetTicket	해당 주문의 티켓 반환
OrderSelect	추가로 작업할 주문을 선택합니다

© 2000-2025, MetaQuotes Ltd.

함수	기능
OrderGetDouble	주문의 요청된 속성을 반환합니다 (double)
OrderGetInteger	주문의 요청된 속성을 반환합니다 (datetime 또는 int)
OrderGetString	주문의 요청된 속성을 반환합니다 (string)
HistorySelect	서버 시간의 지정된 기간 동안 트랜잭션 및 주문 내역을 검색합니다
HistorySelectByPosition	지정된 포지션 식별자를 사용한 거래 내역을 요청합니다.
HistoryOrderSelect	내역에서 추가로 작업할 주문을 선택합니다
HistoryOrdersTotal	내역의 주문 수를 반환합니다
HistoryOrderGetTicket	내역에서 해당 주문의 주문 티켓 반환
HistoryOrderGetDouble	내역에서 주문의 요청된 속성을 반환합니다 (double)
HistoryOrderGetInteger	내역에서 주문의 요청된 속성을 반환합니다 (datetime 또는 int)
HistoryOrderGetString	내역에서 주문의 요청된 속성을 반환합니다 (string)
HistoryDealSelect	내역에서 적절한 함수를 통해 추가로 호출할 거래를 선택합니다
HistoryDealsTotal	내역 내 거래 수를 반환합니다
HistoryDealGetTicket	내역에서 해당 거래의 티켓을 반환합니다
HistoryDealGetDouble	내역에서 거래의 요청된 속성을 반환합니다 (double)
HistoryDealGetInteger	내역에서 거래의 요청된 속성을 반환합니다 (datetime 또는 int)
HistoryDealGetString	내역에서 거래의 요청된 속성을 반환합니다 (string)

© 2000-2025, MetaQuotes Ltd.

OrderCalcMargin

이 함수는 현재 보류 주문 및 오픈 포지션을 고려하지 않고 현재 시장 환경에서 지정된 주문 유형에 필요한 마진을 계산합니다. 계획된 거래 작업에 대한 마진을 평가할 수 있습니다. 값은 계정의 통화로 반환됩니다.

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,           // 주문 유형
   string                symbol,           // 심볼명
   double                volume,           // 양
   double                price,            // 시가
   double&               margin            // 마진 값을 얻기 위한 변수
   );
```

매개 변수

action

[in] 이 값은 ENUM_ORDER_TYPE 열거값 중 하나일 수 있습니다.

Symbol

[in] 심볼명.

volume

[in] 거래 작업의 양.

price

[in] 시가.

margin

[out] 함수가 성공적으로 실행되는 경우 필요한 마진 값이 기록되는 변수. 연산은 현재 계정에서 보류 주문과 오픈 포지션이 없었던 것처럼 수행됩니다. 마진 값은 여러 요소에 따라 달라지며 시장 환경에 따라 다를 수 있습니다.

반환값

이 함수는 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 , GetLastError() 함수를 호출하십시오.

예:

```
#define   VOLUME     1.0   // 주문 볼륨
#define   DEVIATION  100   // distance for setting a pending order
#define   STOP_LIMIT 50    // order StopLimit distance

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   int array_types[8]={ORDER_TYPE_BUY,
                       ORDER_TYPE_SELL,
```

© 2000-2025, MetaQuotes Ltd.

```
                        ORDER_TYPE_BUY_LIMIT,
                        ORDER_TYPE_SELL_LIMIT,
                        ORDER_TYPE_BUY_STOP,
                        ORDER_TYPE_SELL_STOP,
                        ORDER_TYPE_BUY_STOP_LIMIT,
                        ORDER_TYPE_SELL_STOP_LIMIT };

//--- 주문 유형 배열의 루프에서
   for(int i=0; i<(int)array_types.Size(); i++)
     {
      //--- 주문 유형에 따라 ORDER_TYPE_BUY 또는 ORDER_TYPE_SELL 유형을 획득
      ENUM_ORDER_TYPE type=MarketOrderByOrderType((ENUM_ORDER_TYPE)array_types[i]);

      //--- 주문 유형에 따라 가격 획득
      double price=PriceByOrderType(_Symbol, type);

      //--- 'action'에 지정된 주문 유형에 필요한 마진 금액을 획득
      double margin=EMPTY_VALUE;
      ResetLastError();
      if(!OrderCalcMargin(type, _Symbol, VOLUME, price, margin))
        {
         Print("OrderCalcMargin() failed. Error ",GetLastError());
         continue;
        }
      //--- 결과를 저널에 인쇄
      PrintFormat("Margin required for %.2f %s order at price %.*f on %s symbol: %.2f
     }
   /*
   결과:
   Margin required for 1.00 Buy order at price 1.31668 on GBPUSD symbol: 1316.68 USD
   Margin required for 1.00 Sell order at price 1.31661 on GBPUSD symbol: 1316.61 USD
   Margin required for 1.00 Buy Limit order at price 1.31568 on GBPUSD symbol: 1315.68
   Margin required for 1.00 Sell Limit order at price 1.31761 on GBPUSD symbol: 1317.6
   Margin required for 1.00 Buy Stop order at price 1.31768 on GBPUSD symbol: 1317.68
   Margin required for 1.00 Sell Stop order at price 1.31561 on GBPUSD symbol: 1315.61
   Margin required for 1.00 Buy Stop Limit order at price 1.31718 on GBPUSD symbol: 13
   Margin required for 1.00 Sell Stop Limit order at price 1.31611 on GBPUSD symbol: 1
   */
  }
//+------------------------------------------------------------------+
//| 매수 또는 매도 주문 유형 반환                                    |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE MarketOrderByOrderType(ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY  : case ORDER_TYPE_BUY_LIMIT  : case ORDER_TYPE_BUY_STOP  :
        return(ORDER_TYPE_BUY);
      case ORDER_TYPE_SELL : case ORDER_TYPE_SELL_LIMIT : case ORDER_TYPE_SELL_STOP :
```

© 2000-2025, MetaQuotes Ltd.

```
         return(ORDER_TYPE_SELL);
      }
    return(WRONG_VALUE);
   }
//+------------------------------------------------------------------+
//| 주문 유형별 주문 진입 가격 반환                                  |
//+------------------------------------------------------------------+
double PriceByOrderType(const string symbol, const ENUM_ORDER_TYPE order_type)
  {
   int     digits=0;
   double  point=0;
   MqlTick tick={};

//--- 심볼 포인트 값 획득
   ResetLastError();
   if(!SymbolInfoDouble(symbol, SYMBOL_POINT, point))
     {
      Print("SymbolInfoDouble() failed. Error ", GetLastError());
      return 0;
     }

//--- 심볼 자리수 획득
   long value=0;
   if(!SymbolInfoInteger(symbol, SYMBOL_DIGITS, value))
     {
      Print("SymbolInfoInteger() failed. Error ", GetLastError());
      return 0;
     }
   digits=(int)value;

//--- 심볼별 마지막 가격 가격 획득
   if(!SymbolInfoTick(symbol, tick))
     {
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return 0;
     }

//--- 주문 유형별 가격 반환
   switch(order_type)
     {
      case ORDER_TYPE_BUY              :  return(tick.ask);
      case ORDER_TYPE_SELL             :  return(tick.bid);
      case ORDER_TYPE_BUY_LIMIT        :  return(NormalizeDouble(tick.ask - DEVIATION
      case ORDER_TYPE_SELL_LIMIT       :  return(NormalizeDouble(tick.bid + DEVIATION
      case ORDER_TYPE_BUY_STOP         :  return(NormalizeDouble(tick.ask + DEVIATION
      case ORDER_TYPE_SELL_STOP        :  return(NormalizeDouble(tick.bid - DEVIATION
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return(NormalizeDouble(tick.ask + DEVIATION
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return(NormalizeDouble(tick.bid - DEVIATION
      default                          :  return(0);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                            |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE order_type)
   {
    switch(order_type)
      {
       case ORDER_TYPE_BUY              :  return("Buy");
       case ORDER_TYPE_SELL             :  return("Sell");
       case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
       case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
       case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
       case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
       case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
       case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
       case ORDER_TYPE_CLOSE_BY         :  return("Close By");
       default                          :  return("Unknown order type");
      }
   }
```

참고 항목

OrderSend(), Order Properties, Trade Operation Types

© 2000-2025, MetaQuotes Ltd.

OrderCalcProfit

이 함수는 전달된 매개 변수를 기반으로 현재 시장 조건에서 현재 계정의 이익을 계산합니다. 이 함수는 거래 작업의 결과를 사전 평가하는 데 사용됩니다. 값은 계정의 통화로 반환됩니다.

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,          // 주문 유형 (ORDER_TYPE_BUY or ORDER_TYPE_S
   string                symbol,          // 심볼명
   double                volume,          // 양
   double                price_open,      // 시가
   double                price_close,     // 종가
   double&               profit           // 수익가치를 얻기 위한 변수
   );
```

매개 변수

action

[in] 주문 유형, ENUM_ORDER_TYPE 열거값의 다음 두 값 중 하나가 될 수 있습니다: ORDER_TYPE_BUY 또는 ORDER_TYPE_SELL.

Symbol

[in] 심볼명.

volume

[in] 거래 작업의 양.

price_open

[in] 시가.

price_close

[in] 종가.

profit

[out] 함수가 성공적으로 실행될 경우 계산된 이익 값이 기록될 변수. 예상 수익 가치는 여러 요소에 따라 달라지며 시장 환경에 따라 다를 수 있습니다.

반환값

이 함수는 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 잘못된 주문 유형을 지정하면 함수가 false를 반환합니다. 오류에 대한 정보를 얻으려면, GetLastError()를 호출하십시오.

예:

```
#define   VOLUME     1.0   // 주문 볼륨
#define   DEVIATION  100   // 종가 전 포인트 수

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   string currency_profit=SymbolInfoString(_Symbol,SYMBOL_CURRENCY_PROFIT);   //수익 통
   double close_price=ChartPriceOnDropped(); //마우스로 스크립트를 떨어뜨린 지점에 해당하는

//---
   for(int i=0; i<=ORDER_TYPE_SELL; i++)
     {
      ENUM_ORDER_TYPE order_type=(ENUM_ORDER_TYPE)i;                  // 주문 유형: 0 -
      double open_price=PriceOpenByOrderType(_Symbol, order_type);    // 시가: 매수의 경

      //--- 시가를 받지 못한 경우 루프를 계속 진행
      if(open_price==0)
         continue;

      //--- 종가가 0인 경우(차트에 드래그하여 스크립트가 실행되지 않은 경우) 가격을 계산
      if(close_price==0)
         close_price=(order_type==ORDER_TYPE_BUY ? open_price + DEVIATION * _Point : c

      //--- 전달된 매개변수를 기반으로 현재 계정 및 시장 환경에 대한 대략적인 이익 규모를 계산
      double profit=0;
      ResetLastError();
      if(!OrderCalcProfit(order_type, _Symbol, VOLUME, open_price, close_price, profit
        {
         Print("OrderCalcProfit() failed. Error ", GetLastError());
         continue;
        }

      //--- 받은 수익 값을 저널에 인쇄
      PrintFormat("Estimated profit for %.2f %s position on %s with opening price of %
                  VOLUME, OrderTypeDescription(order_type), _Symbol, _Digits, open_pri

     }
   /*
   결과:
   Estimated profit for 1.00 Buy position on EURUSD with opening price of 1.10757 and
   Estimated profit for 1.00 Sell position on EURUSD with opening price of 1.10753 and
   */
  }
//+------------------------------------------------------------------+
//| 주문 유형별 주문 진입 가격 반환                                  |
//+------------------------------------------------------------------+
double PriceOpenByOrderType(const string symbol, const ENUM_ORDER_TYPE order_type)
  {
   MqlTick tick={};

//--- 심볼별 마지막 가격 가격 획득
   if(!SymbolInfoTick(symbol, tick))
     {
      Print("SymbolInfoTick() failed. Error ", GetLastError());
      return 0;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- 주문 유형별 가격 반환
   switch(order_type)
      {
       case ORDER_TYPE_BUY              :  return(tick.ask);
       case ORDER_TYPE_SELL             :  return(tick.bid);
       default                          :  return(0);
      }
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                              |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE order_type)
  {
   switch(order_type)
      {
       case ORDER_TYPE_BUY              :  return("Buy");
       case ORDER_TYPE_SELL             :  return("Sell");
       default                          :  return("Unknown order type");
      }
  }
```

참고 항목

OrderSend(), Order Properties, Trade Operation Types

© 2000-2025, MetaQuotes Ltd.

OrderCheck

OrderCheck() 함수는 필요한 거래 작업을 실행할 충분한 자금이 있는지 확인합니다. 검사 결과는 MqlTradeCheckResult 구조의 필드에 배치됩니다.

```
bool  OrderCheck(
   MqlTradeRequest&        request,      // 요청 구조
   MqlTradeCheckResult&    result        // 결과 구조
   );
```

매개 변수

요청

[in] 필요한 거래 작업을 설명하는 MqlTradeRequest 유형의 구조에 대한 포인터.

결과

[in,out] 검사 결과가 배치될 MqlTradeCheckResult 유형의 구조에 대한 포인터.

반환값

자금이 작업에 충분하지 않거나 파라미터가 잘못 입력되면 함수는 False를 반환합니다. 구조의 기본 검사(포인터 검사)에 성공하면 true를 반환합니다. **그러나 이것은 요청된 거래 작업이 성공적으로 실행될 것이라는 징후는 아닙니다.**. 함수 실행 결과에 대한 자세한 설명은 *결과* 구조 필드를 분석하십시오.

오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출하십시오.

예:

```
#define   DEVIATION     5              // 가격에서 허용된 편차
#define   VOLUME        1.0            // 주문 볼륨
#define   EXPERT_MAGIC  123            // MagicNumber
#define   DIRECTION     ORDER_TYPE_BUY // 진입한 가격 방향 (ORDER_TYPE_BUY or ORDER_TYPE

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 요청, 확인 및 결과 구조를 선언하고 초기화
   MqlTradeRequest     request={};
   MqlTradeCheckResult check  ={};
   MqlTradeResult      result ={};

//---거래 요청 매개변수 준비
   PrepareRequest(_Symbol, DIRECTION, VOLUME, request);

//--- 거래 요청 매개변수 확인
   ResetLastError();
   bool res=OrderCheck(request, check);
   if(!res)
     {
      PrintFormat("Trade request verification completed with error %d\nServer retcode
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- 거래 요청 확인이 성공했습니다 - 거래 요청 확인 구조 필드에 대한 설명을 표시
   Print("Trade request verification completed successfully");
   MqlTradeCheckResultPrint(check, 14);

//--- 거래 요청 전송
   if(!OrderSend(request, result))
      Print("OrderSend error ", GetLastError());    // 요청을 보낼 수 없으면 오류 코드를 

//--- 운용에 대한 정보
   PrintFormat("Trade request result: retcode=%u, deal=%I64u, order=%I64u", result.ret
   /*
   클라이언트 터미널에서 자동 거래가 비활성화 된 결과
   Trade request verification completed with error 4752
   Server retcode: 10027, comment: AutoTrading disabled by client

   자동 거래를 활성화하고 폐장한 시장에서 다시 확인
   Experts   automated trading is enabled
   Trade request verification completed successfully
   Retcode:      0
   Balance:      10779.50 USD
   Equity:       10779.50 USD
   Profit:       0.00 USD
   Margin:       1104.79 USD
   Margin free:  9674.71 USD
   Margin level: 975.71 %
   Comment:      Done
   OrderSend error 4756
   Trade request result: retcode=10018, deal=0, order=0

   시장에서 확인
   Trade request verification completed successfully
   Retcode:      0
   Balance:      10779.50 USD
   Equity:       10779.50 USD
   Profit:       0.00 USD
   Margin:       110.46 USD
   Margin free:  10669.04 USD
   Margin level: 9758.74 %
   Comment:      Done
   Trade request result: retcode=10009, deal=2777010968, order=2802818813
   */
  }
//+------------------------------------------------------------------+
//| 거래 요청을 위한 매개변수 준비                                   |
//+------------------------------------------------------------------+
void PrepareRequest(const string symbol, const ENUM_ORDER_TYPE order_type, const doubl
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   ENUM_ORDER_TYPE type=(DIRECTION !=ORDER_TYPE_BUY ? ORDER_TYPE_SELL : DIRECTION);
   double price=(DIRECTION==ORDER_TYPE_BUY ? SymbolInfoDouble(Symbol(), SYMBOL_ASK) :
//--- 매개 변수 요청
   request.action    = TRADE_ACTION_DEAL; // 거래 운영 유형
   request.symbol    = symbol;            // 심볼
   request.volume    = volume;            // 볼륨
   request.type      = type;              // 주문 유형
   request.price     = price;             // 시가
   request.deviation = DEVIATION;         // 가격에서 허용된 편차
   request.magic     = EXPERT_MAGIC;      // 주문 매직넘버
  }
//+------------------------------------------------------------------+
//| 거래 요청 필드 출력                                                |
//| 저널에서 결과 검증                                                 |
//+------------------------------------------------------------------+
void MqlTradeCheckResultPrint(const MqlTradeCheckResult &check, const uint header_widt
  {
//--- 계좌 통화와 계좌 통화의 소수 자릿수를 획득
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
   int    digits  =(int)AccountInfoInteger(ACCOUNT_CURRENCY_DIGITS);

//--- 헤더 텍스트와 헤더 필드의 너비를 정의합니다.
//--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 10이 됩니다.
   string header="Retcode:";
   uint w=(header_width==0 ? header.Length()+1 : header_width);
//--- 저널에 지정된 너비의 헤더와 함께 반환 코드를 표시
   PrintFormat("%-*s%-u", w, header, check.retcode);

//--- 저널에서 거래 작업을 실행한 후 잔액 값을 표시
   header="Balance:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.balance, currency);

//--- 저널에서 거래 작업을 실행한 후 자산 가치를 표시
   header="Equity:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.equity, currency);

//--- 저널에서 거래 작업을 실행한 후 유동 이익 값을 표시
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.profit, currency);

//--- 필요한 거래 작업에 필요한 마진 금액을 저널에 표시
   header="Margin:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.margin, currency);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 거래 작업을 수행한 후 남은 자산 가치를 저널에 표시
   header="Margin free:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.*f %s", w, header, digits, check.margin_free, currency);

//--- 저널에서 필요한 거래 작업을 완료한 후 설정할 마진 수준을 표시
   header="Margin level:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-.2f %%", w, header, check.margin_level);

//--- 응답 코드에 대한 코멘트와 오류 설명을 저널에 표시
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   PrintFormat("%-*s%-s", w, header, check.comment);
  }
```

참고 항목

OrderSend(), Trade Operation Types, Trade Request Structure, Structure of Request Check Results, Structure of a Trade Request Result

© 2000-2025, MetaQuotes Ltd.

OrderSend

OrderSend() 함수는 거래 작업을 실행하는 데 요청을 거래 서버로 전송하도록 사용됩니다.

```
bool  OrderSend(
   MqlTradeRequest&  request,      // 쿼리 구조
   MqlTradeResult&   result        // 답변 구조
   );
```

Parameter

request

[in] 클라이언트의 거래 활동을 설명하는 MqlTradeRequest 유형의 구조에 대한 포인터.

result

[in,out] 완료 시 거래 작업의 결과를 설명하는 MqlTradeResult 유형의 구조에 대한 포인터(참이 반환되는 경우).

반환값

구조의 기본 검사(지수 검사)에 성공하면 true를 반환합니다. **그러나, 이것은 거래 작업의 성공적인 실행의 징후는 아닙니다.**. 함수 실행 결과에 대한 자세한 설명은 *결과* 구조의 필드를 분석합니다.

참고

거래 요청은 거래 서버에서 확인하는 여러 단계를 거칩니다. 먼저 *요청* 매개 변수의 모든 필수 필드가 올바르게 작성되었는지 확인합니다. 오류가 없으면 서버는 추가 처리를 위해 주문을 수락합니다. 거래 서버에서 주문이 성공적으로 수락되면 OrderSend() 함수가 true를 반환합니다.

거래 서버로 전송하기 전에 요청을 확인하는 것이 좋습니다. 요청을 확인하려면, OrderCheck() 함수를 사용하십시오. 또한 거래 운영을 실행하기에 충분한 자금이 있는지 확인하고 거래 요청 확인 결과에 다음과 같은 많은 유용한 매개 변수를 반환합니다:

- 반환 코드(선택한 요청의 오류에 대한 정보를 포함);
- 거래 작업 실행 후 나타나는 잔액 값;
- 거래 작업 실행 후 나타날 순 가치;
- 거래 작업 실행 후 표시될 floating 소수점 값;
- 거래 작업에 요구되는 마진;
- 거래 작업 실행 후 유지되는 자유 자본의 금액;
- 거래 작업 실행 후 설정될 마진 수준;
- 응답 코드에 대한 설명, 오류 설명에 대한 코멘트.

마켓 오더(MqlTradeRequest.action=TRADE_ACTION_DEAL)을 보낼 때, OrderSend() 함수의 성공적 결과가 꼭 주문이 실행(적절한 거래 수행)되었다는 것을 의미하지는 않습니다. 이 경우, 'true'는 추가적 실행을 위해 거래 시스템에 성공적으로 주문되었음을 의미합니다. 거래 서버는 *딜 또는 주문* 필드 값을 결과 구조에서 채울 수 있는데, OrderSend() 호출에 응답할 때 이러한 데이터를 알고 있는 경우에 그러합니다. 일반적으로 주문에 해당하는 거래를 실행하는 이벤트는 OrderSend() 호출에 응답을 보낸 후에 발생할 수 있습니다. 따라서 어떠한 거래 요청 유형에서도 OrderSend() 실행 결과를 받을 때, *retcode* 거래 서버 응답 코드와 *retcode_external* 외부 시스템 응답 코드 (필요시)를 먼저 확인해야 하며, 이는 획득한 결과 구조에서 사용 가능합니다.

수락된 각 주문은 실행 조건 중 하나가 발생할 때까지 처리 대기 중인 거래 서버에 저장됩니다:

- 만료,
- 반대의 청탁,

© 2000-2025, MetaQuotes Ltd.

- 실행 가격이 표시될 때 실행을 명령,
- 주문 취소 요청 접수.

주문 처리 순간, 거래 서버는 Trade 이벤트 발생에 대한 메시지를 단말기에 전송하며, 이는 OnTrade() 함수로 처리할 수 있습니다.

OrderSend() 함수로 전송된 서버에서 거래 요청을 실행한 결과는 OnTradeTransaction 핸들러로 추적할 수 있습니다. OnTradeTransaction 핸들러는 하나의 거래 요청을 실행할 때 여러 번 호출됩니다.

예를 들어, 시장 구매 주문서를 보낼 때, 주문이 처리되고, 계정에 대해 적절한 구매 주문서가 생성되고, 주문이 실행되어 보류 주문 목록에서 제거되며, 주문 기록에 적절한 거래가 추가되고, 새 포지션이 생성됩니다. 이러한 각 이벤트에 대해 OnTradeTransaction 함수가 호출됩니다.

예:

```
//--- ORDER_MAGIC 값
input long order_magic=55555;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 계정이 데모인지 확인
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("라이브 계정에서는 스크립트 작업이 허용되지 않습니다!");
      return;
     }
//--- 주문 또는 주문 삭제
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- no current orders - place an order
      uint res=SendRandomPendingOrder(order_magic);
      Print("거래 서버의 반환 코드 ",res);
     }
   else // 주문 - 주문 삭제 가 있습니다
     {
      DeleteAllOrdersByMagic(order_magic);
     }
//---
  }
//+------------------------------------------------------------------+
//| 지정된 ORDER_MAGIC가 있는 현재 주문 수를 수신합니다              |
//+------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
   ulong order_ticket;
   int total=0;
//--- 모든 보류 주문 살펴보기
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
```

© 2000-2025, MetaQuotes Ltd.

```
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------+
//| 지정된 ORDER_MAGIC을 사용하여 보류 주문을 모두 삭제                |
//+------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- 모든 보류 주문 살펴보기
   for(int i=OrdersTotal()-1;i>=0;i--)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- 적합한 ORDER_MAGIC로 주문
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={0};
            MqlTradeRequest request={0};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- 로그에 서버 응답을 기록
            Print(__FUNCTION__,": ",result.comment," reply code ",result.retcode);
           }
//---
  }
//+------------------------------------------------------------------+
//| 보류 중인 순서를 임의 방식으로 설정                               |
//+------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- 요구 준비
   MqlTradeRequest request={0};
   request.action=TRADE_ACTION_PENDING;          // 보류 주문 설정
   request.magic=magic_number;                   // ORDER_MAGIC
   request.symbol=_Symbol;                       // symbol
   request.volume=0.1;                           // 볼륨 0.1 롯
   request.sl=0;                                 // 스탑 로스가 명시되지 않았습니다
   request.tp=0;                                 // 이익 실현이 명시되지 않았습니다
//--- 주문 유형 형성
   request.type=GetRandomType();                 // 주문 유형
//--- 보류 주문에 대한 가격 형성
   request.price=GetRandomPrice(request.type);   // open price
//--- 거래 요청 전송
   MqlTradeResult result={0};
   OrderSend(request,result);
//--- 로그에 서버 응답을 기록
   Print(__FUNCTION__,":",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 거래 서버 응답의 반환 코드
   return result.retcode;
  }
//+------------------------------------------------------------------+
//| 보류 주문의 유형을 임의 방식으로 반환                                |
//+------------------------------------------------------------------+
ENUM_ORDER_TYPE GetRandomType()
  {
   int t=MathRand()%4;
//---   0<=t<4
   switch(t)
     {
      case(0):return(ORDER_TYPE_BUY_LIMIT);
      case(1):return(ORDER_TYPE_SELL_LIMIT);
      case(2):return(ORDER_TYPE_BUY_STOP);
      case(3):return(ORDER_TYPE_SELL_STOP);
     }
//--- 부정확한 값
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//| 임의의 방법으로 가격 반환                                          |
//+------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- 심볼에 대한 스탑 레벨
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- 최근 틱 데이터 수신
   MqlTick last_tick={0};
   SymbolInfoTick(_Symbol,last_tick);
//--- 유형에 따라 가격 계산
   double price;
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT or ORDER_TYPE_SELL_STOP
     {
      price=last_tick.bid; // Bid 가격에서부터 시작
      price=price-(distance+(MathRand()%10)*5)*_Point;
     }
   else             // ORDER_TYPE_SELL_LIMIT 또는 ORDER_TYPE_BUY_STOP
     {
      price=last_tick.ask; // Ask 가격에서부터 시작
      price=price+(distance+(MathRand()%10)*5)*_Point;
     }
//---
   return(price);
  }
```

참고 항목

© 2000-2025, MetaQuotes Ltd.

Trade Operation Types, Trade Request Structure, Structure of Request Check Results, Structure of a Trade Request Result

© 2000–2025, MetaQuotes Ltd.

OrderSendAsync

OrderSendAsync() 함수는 비동기 거래 작업작업을 수행하는 데 사용되며, 이 때 전송된 요청에 대한 거래 서버의 응답을 기다리지 않습니다. 이 함수는 거래 알고리즘 조건에 따라 서버의 응답을 기다리는 데 시간을 낭비할 수 없는 고주파 거래를 위해 설계되었습니다.

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,      // 요청 구조
   MqlTradeResult&   result        // 반응 구조
   );
```

Parameter

request

[in] 클라이언트의 거래 작업을 설명하는 MqlTradeRequest 유형의 구조에 대한 포인터.

result

[in,out] 함수 실행 시 (true가 반환되는 경우) 거래 작업의 결과를 설명하는 MqlTradeResult 유형의 구조에 대한 포인터.

반환값

요청이 거래 서버로 전송되면 true를 반환합니다.. 요청이 전송되지 않으면 false를 반환합니다. 요청이 전송된 경우 *결과* 변수에서 응답 코드는 TRADE_RETCODE_PLACED 값(코드 10008) - "주문 발주됨". 실행에 성공하면 전송 사실만 의미할 뿐, 요청이 거래 서버에 도달하여 처리가 수락되었다는 보장은 없습니다. 요청을 처리할 때 거래 서버는 클라이언트 터미널에 현재 위치, 주문 및 거래 상태의 변경을 통지하여 트레이드 이벤트가 생성되도록 합니다.

OrderSendAsync() 함수로 보낸 서버에서 거래요청 실행 결과는 OnTradeTransaction 핸들러로 추적 가능합니다. OnTradeTransaction 핸들러는 하나의 거래 요청을 실행할 때 여러 번 호출됩니다.

예를 들어, 시장 구매 주문서를 보낼 때, 주문이 처리되고, 계정에 대해 적절한 구매 주문서가 생성되고, 주문이 실행되어 보류 주문 목록에서 제거되며, 주문 기록에 적절한 거래가 추가되고, 새 포지션이 생성됩니다. 이러한 각 이벤트에 대해 OnTradeTransaction 함수가 호출됩니다. 이러한 데이터를 얻으려면 함수 파라미터를 분석해야 합니다.

- **trans** - 이 매개 변수는 거래 계정에 적용되는 트레이드 트랜잭션을 설명하는 MqlTradeTransaction 구조를 취합니다;
- **request** - 이 매개 변수는 트레이드 트랜잭션을 초래한 거래 요청을 설명하는 MqlTradeRequest 구조를 취합니다;
- **result** - 이 매개 변수는 거래 요청 실행 결과를 설명하는 MqlTradeResult 구조를 가져옵니다.

참고

목적과 매개 변수 면에서, 함수는 OrderSend()와 유사하지만, 그것과는 달리 비동기적입니다. 즉, 함수 실행 결과를 기다리는 동안 프로그램 작동을 보류하지 않습니다. 샘플 Expert Advisor를 사용하여 이 두 기능의 거래 비율을 비교할 수 있습니다.

예:

```
#property description "거래 요청 전송을 위한 Expert Advisor "
                      " OrderSendAsync() 함수를 이용한 ₩₩"
#property description "다음을 사용하여 거래 이벤트를 핸들링"
                      " OnTrade() 및 OnTradeTransaction() 핸들러 함수가 표시됩니다₩₩"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "Expert Advisor 매개 변수는 Magic Number 설정을 허용합니다"
                      " (unique ID) "
#property description "그리고 전문가 로그에 메시지를 표시하는 모드. 모든 세부 정보가 기본적으로
//--- 입력 매개 변수
input int  MagicNumber=1234567;      // Expert Advisor ID
input bool DescriptionModeFull=true; // 상세 출력 모드
//--- HistorySelect() 호출에 사용하기 위한 변수
datetime history_start;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 오토트레이딩이 가능한지 확인
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("터미널 내 오토트레이딩이 비활성화되면 Expert Advisor가 제거됩니다.");
      ExpertRemove();
      return(-1);
     }
//--- 실제 계정에서 거래할 수 없음
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Expert Advisor는 실제 계정으로 거래할 수 없습니다!");
      ExpertRemove();
      return(-2);
     }
//--- 이 계정으로 거래할 수 있는지 확인(예: 투자자 비밀번호를 사용하는 경우 거래 불가).
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("이 계정에서의 거래는 불가능합니다");
      ExpertRemove();
      return(-3);
     }
//--- 거래 내역 수신을 위한 Expert Advisor의 런칭 시간을 저장
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 취소 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 모든 그래픽 개체 삭제
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| TradeTransaction 함수                                            |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- 거래 이벤트의 핸들러 기능의 이름을 딴 헤더
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- 열거값으로 트랜잭션 유형 수신
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- 거래가 요청 처리의 결과인 경우
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //--- 거래명 표시
      Print(EnumToString(type));
      //--- 그런 다음 처리된 요청에 대한 문자열 설명을 표시합니다.
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- 요청 결과에 대한 설명 표시
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else // 다른 유형의 트랜잭션에 대한 전체 설명을 표시합니다
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------+
//| 거래 함수                                                        |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- 거래 계정 상태를 저장하기 위한 정적 멤버
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
//--- 거래 내역 요청
   bool update=HistorySelect(history_start,TimeCurrent());
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)update);
//--- 거래 이벤트의 핸들러 기능의 이름을 딴 헤더
   Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- 핸들러 이름 및 처리 당시의 주문 수를 표시합니다
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 주문 수, 포지션, 거래 및 변경 사항을 괄호 안에 표시합니다
   PrintFormat("PositionsTotal() = %d (%+d)",
               curr_positions,(curr_positions-prev_positions));
   PrintFormat("OrdersTotal() = %d (%+d)",
               curr_orders,curr_orders-prev_orders);
   PrintFormat("HistoryOrdersTotal() = %d (%+d)",
               curr_deals,curr_deals-prev_deals);
   PrintFormat("HistoryDealsTotal() = %d (%+d)",
               curr_history_orders,curr_history_orders-prev_history_orders);
//--- 보다 편리하게 로그를 볼 수 있도록 줄 바꿈 삽입
   Print("");
//--- 계정 상태 저장
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- CHARTEVENT_CLICK 이벤트 핸들링 ("차트 클릭")
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //--- 딜의 최소량
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //--- "매수" 버튼이 눌린다면 매수합니다
      if(sparam=="Buy")
        {
         PrintFormat("Buy %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- 버튼 누름취소
         ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
      //--- "매도" 버튼이 눌렸다면 매도합니다
      if(sparam=="Sell")
        {
         PrintFormat("Sell %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
         //--- 버튼 누름취소
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//---
  }
//+------------------------------------------------------------------+
//| 트랜잭션의 텍스트 설명을 반환합니다                                |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- 함수로부터 반환될 수 있는 문자열 준비
   string desc=EnumToString(trans.type)+"\r\n";
//--- 가능한 모든 데이터가 상세 모드로 추가됨
   if(detailed)
     {
      desc+="Symbol: "+trans.symbol+"\r\n";
      desc+="Deal ticket: "+(string)trans.deal+"\r\n";
      desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Order ticket: "+(string)trans.order+"\r\n";
      desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
      desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
      desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
      desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
      desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
      desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
     }
//--- 받은 문자열 반환
   return desc;
  }
//+------------------------------------------------------------------+
//| 거래 요청에 대한 텍스트 설명을 반환합니다                          |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- 함수로부터 반환될 수 있는 문자열 준비
   string desc=EnumToString(request.action)+"\r\n";
//--- 세부 모드에서 사용 가능한 모든 데이터 추가
   if(detailed)
     {
      desc+="Symbol: "+request.symbol+"\r\n";
      desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Order ticket: "+(string)request.order+"\r\n";
      desc+="Order type: "+EnumToString(request.type)+"\r\n";
      desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
      desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
      desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
```

© 2000-2025, MetaQuotes Ltd.

```
      desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
      desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
      desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
      desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
      desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
      desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
      desc+="Comment: "+request.comment+"\r\n";
     }
//--- 받은 문자열 반환
   return desc;
  }
//+------------------------------------------------------------------+
//| 요청 처리 결과의 텍스트 설명을 반환합니다                                  |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- 함수로부터 반환된 문자열 준비
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- 세부 모드에서 사용 가능한 모든 데이터 추가
   if(detailed)
     {
      desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
      desc+="Order ticket: "+(string)result.order+"\r\n";
      desc+="Deal ticket: "+(string)result.deal+"\r\n";
      desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
      desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Comment: "+result.comment+"\r\n";
     }
//--- 받은 문자열 반환
   return desc;
  }
//+------------------------------------------------------------------+
//| 구매 및 판매를 위한 두 가지 버튼 생성                                    |
//+------------------------------------------------------------------+
void CreateBuySellButtons()
  {
//--- 이름이 "Buy"인 개체 확인
   if(ObjectFind(0,"Buy")>=0)
     {
      //--- 발견된 개체가 버튼이 아닌 경우 삭제
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Buy");
     }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // "Buy" 버튼 생성
//--- "Buy" 버튼 구성
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectSetInteger(0,"Buy",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Buy",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Buy",OBJPROP_YDISTANCE,50);
   ObjectSetInteger(0,"Buy",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Buy",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Buy",OBJPROP_TEXT,"Buy");
   ObjectSetInteger(0,"Buy",OBJPROP_COLOR,clrRed);
//--- 이름이 "Sell"인 개체의 존재 여부 확인
   if(ObjectFind(0,"Sell")>=0)
     {
      //--- 발견된 개체가 버튼이 아닌 경우 삭제
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); // "Sell" 버튼 생성
//--- "Sell" 버튼 구성
   ObjectSetInteger(0,"Sell",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
   ObjectSetInteger(0,"Sell",OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,"Sell",OBJPROP_XSIZE,70);
   ObjectSetInteger(0,"Sell",OBJPROP_YSIZE,30);
   ObjectSetString(0,"Sell",OBJPROP_TEXT,"Sell");
   ObjectSetInteger(0,"Sell",OBJPROP_COLOR,clrBlue);
//--- 즉시 버튼을 표시하도록 차트 강제 업데이트를 수행합니다
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------+
//| OrderSendAsync() 비동기 함수를 사용하여 구매                     |
//+------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- 요청 준비
   MqlTradeRequest req={0};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_BUY;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
   req.deviation   =10;
   req.comment     ="Buy using OrderSendAsync()";
   MqlTradeResult  res={0};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": error ",GetLastError(),", retcode = ",res.retcode);
     }
//---
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 비동기식 OrderSendAsync() 함수를 사용하여 판매                    |
//+------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- 요청 준비
   MqlTradeRequest req={0};
   req.action      =TRADE_ACTION_DEAL;
   req.symbol      =_Symbol;
   req.magic       =MagicNumber;
   req.volume      =0.1;
   req.type        =ORDER_TYPE_SELL;
   req.price       =SymbolInfoDouble(req.symbol,SYMBOL_BID);
   req.deviation   =10;
   req.comment     ="Sell using OrderSendAsync()";
   MqlTradeResult  res={0};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": error ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
```

"Experts" 로그에 메시지 표시 예제:

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnChartEvent: sparam = Sell
12:52:52   ExpertAdvisor (EURUSD,H1)   Sell EURUSD 0.01 lot
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_REQUEST
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------RequestDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_ACTION_DEAL
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Magic Number: 1234567
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order filling: ORDER_FILLING_FOK
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Deviation points: 10
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Limit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment: Sell using OrderSendAsync()
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ ResultDescription
```

© 2000-2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Retcode 10009
12:52:52   ExpertAdvisor (EURUSD,H1)   Request ID: 2
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Ask: 1.29319
12:52:52   ExpertAdvisor (EURUSD,H1)   Bid: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Comment:
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+1)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+2)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_ORDER_DELETE
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
```

© 2000-2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_HISTORY_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_FILLED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTradeTransaction at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   ------------ TransactionDescription
12:52:52   ExpertAdvisor (EURUSD,H1)   TRADE_TRANSACTION_DEAL_ADD
12:52:52   ExpertAdvisor (EURUSD,H1)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal ticket: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Deal type: DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Order ticket: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   Order type: ORDER_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Order state: ORDER_STATE_STARTED
12:52:52   ExpertAdvisor (EURUSD,H1)   Order time type: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Order expiration: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Price: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Price trigger: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Stop Loss: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Take Profit: 0
12:52:52   ExpertAdvisor (EURUSD,H1)   Volume: 0.1
```

© 2000-2025, MetaQuotes Ltd.

```
12:52:52   ExpertAdvisor (EURUSD,H1)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistorySelect( 09:34 , 09:52) = true
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
```

© 2000-2025, MetaQuotes Ltd.

PositionsTotal

오픈 포지션 수를 반환합니다.

```
int  PositionsTotal();
```

반환값

int 유형의 값.

참고

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 심볼 하나에 대해 여러 포지션을 오픈할 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 저널의 계정에서 진입한 포지션 수를 가져오고 출력합니다.
   int total=PositionsTotal();
   Print("Number of open positions on account: ", total);
   /*
  결과:
   Number of open positions on account: 2
   */
  }
```

참고 항목

PositionGetSymbol(), PositionSelect(), Position Properties

© 2000-2025, MetaQuotes Ltd.

PositionGetSymbol

오픈 포지션에 해당하는 심볼을 반환하고 PositionGetDouble, PositionGetInteger, PositionGetString 함수를 사용하여 추가로 작업할 포지션을 자동으로 선택합니다.

```
string  PositionGetSymbol(
   int  index      // 포지션 목록에서의 번호
   );
```

매개 변수

index

[in] 오픈 포지션 목록에서의 포지션 번호.

반환값

string 유형의 값. 포지션을 찾을 수 없으면 빈 문자열이 반환됩니다. To get an error code, call the GetLastError() function.

참고

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 심볼 하나에 대해 여러 포지션을 오픈할 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 계좌의 오픈 포지션 수를 가져옵니다.
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- i 루프 인덱스로 포지션 심볼 가져오기
      ResetLastError();
      string symbol=PositionGetSymbol(i);

      //--- 포지션 심볼이 성공적으로 수신되면 i 인덱스의 포지션이 자동으로 선택됩니다.
      //--- 그리고 PositionGetDouble, PositionGetInteger 및 PositionGetString을 사용하여
      if(symbol!="")
        {
         ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE)
         PrintFormat("Position symbol at index %d: %s, position type: %s", i, symbol,
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          PrintFormat("PositionGetSymbol(%d) failed. Error %d", i, GetLastError());
          continue;
         }
      }
   /*
   결과:
   Position symbol at index 0: GBPUSD, position type: SELL
   Position symbol at index 1: EURUSD, position type: BUY
   */
  }
```

참고 항목

PositionsTotal(), PositionSelect(), Position Properties

© 2000-2025, MetaQuotes Ltd.

PositionSelect

추가 작업을 위해 오픈 포지션을 선택합니다. 함수가 성공적으로 완료되면 true를 반환합니다. 실패 시 False를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError()를 호출하십시오.

```
bool  PositionSelect(
   string  symbol      // 심볼명
   );
```

매개 변수

Symbol

[in] 금융증권의 이름.

반환값

bool 유형의 값.

참고

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다. 이 경우 PositionSelect는 가장 낮은 포지션의 티켓을 선택합니다.

Function PositionSelect()는 위치에 대한 데이터를 프로그램 환경으로 복사하고 PositionGetDouble(), PositionGetInteger() 및 PositionGetString()를 추가로 호출하면 이전에 복사된 데이터가 반환됩니다. 즉, 포지션 자체는 더 이상 존재하지 않을 수 있지만(또는 볼륨, 방향 등이 변경됨), 이 포지션의 데이터는 여전히 얻을 수 있습니다. 포지션에 대한 새로운 데이터를 수신하려면 해당 포지션을 참조하기 전에 PositionSelect()를 호출하는 것이 좋습니다.

예:

```
#define   SYMBOL_NAME   "EURUSD"

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 지정된 심볼에서 포지션을 선택
   if(!PositionSelect(SYMBOL_NAME))
     {
      PrintFormat("PositionSelect(%s) failed. Error %d",SYMBOL_NAME, GetLastError());
      return;
     }

//--- 포지션이 선택되면 PositionGetDouble(), PositionGetInteger() 및 PositionGetString()
//--- 선택된 포지션 티켓 획득
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
   long ticket=PositionGetInteger(POSITION_TICKET);
   if(ticket==0)
     {
      PrintFormat("Failed to get %s position ticket. Error %d", SYMBOL_NAME, GetLastEr
      return;
     }

//--- 티켓이 성공적으로 수신되면 선택한 포지션 심볼과 티켓을 저널에 출력합니다.
   PrintFormat("The position that is selected on the %s symbol has ticket %I64d", SYMB
   /*
   결과:
   The position that is selected on the EURUSD symbol has ticket 2810846623
   */
  }
```

참고 항목

PositionGetSymbol(), PositionsTotal(), Position Properties

© 2000-2025, MetaQuotes Ltd.

PositionSelectByTicket

포지션에 지정된 티켓 번호를 기준으로 작업할 오픈 포지션을 선택합니다. 성공하면 true를 반환합니다. 함수가 실패하면 false를 반환합니다. 오류 세부정보를 위해서는 GetLastError()를 호출하십시오.

```
bool  PositionSelectByTicket(
   ulong   ticket     // 포지션 티켓
   );
```

매개 변수

ticket

[in] 포지션 티켓.

반환값

bool 유형의 값.

참고

PositionSelectByTicket() 함수는 포지션 데이터를 프로그램 환경에 복사합니다. PositionGetDouble(), PositionGetInteger() 및 PositionGetString()를 추가로 호출하면 이전에 복사된 데이터가 반환됩니다. 포지션이 아직 없는 경우(또는 그 크기, 방향 등이 변경된 경우)에도 마찬가지이며, 데이터가 수신되는 경우가 있습니다. 유효한 포지션 데이터를 수신하려면 데이터에 액세스하기 전에 PositionSelectByTicket() 를 호출하는 것이 좋습니다.

예:

```
#define   EXPERT_MAGIC  123456   // MagicNumber

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 요청 및 결과 구조를 선언하고 초기화
   MqlTradeRequest request={};
   MqlTradeResult  result ={};

//--- 매수 포지션에 진입하려면 거래 요청 매개변수를 입력
   request.action    = TRADE_ACTION_DEAL;                       // 거래 동작 유형
   request.symbol    = Symbol();                                // 심볼
   request.volume    = 0.1;                                     // 0.1 랏의 볼륨
   request.type      = ORDER_TYPE_BUY;                          // 주문 유형
   request.price     = SymbolInfoDouble(Symbol(), SYMBOL_ASK); // 시가
   request.deviation = 5;                                       // 가격에서 허용된 편차
   request.magic     = EXPERT_MAGIC;                            // 주문 매직넘버

//--- 요청 전송 요청 전송에 실패한 경우 오류 코드를 표시하고 작업을 완료
   if(!OrderSend(request, result))
     {
      PrintFormat("OrderSend error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- 작동 데이터 표시
   PrintFormat("Trade request result: retcode: %u, deal: %I64u, order: %I64u", result.

//--- 거래 수행 결과에서 포지션 티켓을 얻어 티켓으로 포지션을 선택
//--- 새로 진입한 포지션의 티켓은 거래를 생성한 주문의 티켓에 해당합니다.
   ulong ticket=result.order;
   ResetLastError();
   if(!PositionSelectByTicket(ticket))
     {
      PrintFormat("PositionSelectByTicket(%I64u) failed. Error %d", ticket, GetLastErr
      return;
     }

//--- 티켓으로 선택한 포지션의 데이터를 저널에 표시
   ENUM_POSITION_TYPE type   = (ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
   long               time   = PositionGetInteger(POSITION_TIME_MSC);
   double             price  = PositionGetDouble(POSITION_PRICE_OPEN);
   double             volume= PositionGetDouble(POSITION_VOLUME);
   string             symbol= PositionGetString(POSITION_SYMBOL);
   int                digits= (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   PrintFormat("Current selected position: %s %.2f %s #%I64u at %.*f, %s",
               symbol, volume, (type==POSITION_TYPE_BUY ? "Buy" : "Sell"), ticket, dig
   /*
   결과:
   Trade request result: retcode: 10009, deal: 2778100901, order: 2803905975
   Current selected position: EURUSD 0.10 Buy #2803905975 at 1.10672, 2024.09.02 12:09
   */
  }
//+------------------------------------------------------------------+
//| 말리세컨드로 시간을 반환                                          |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 3
  }
```

참고 항목

PositionGetSymbol(), PositionsTotal(), Position Properties

© 2000-2025, MetaQuotes Ltd.

PositionGetDouble

이 함수는 PositionGetSymbol 또는 PositionSelect를 사용하여 사전 선택된 오픈 포지션의 요청된 속성을 반환합니다. 위치 속성은 double 유형이어야 합니다. 이 함수엔 2가지 변수가 있습니다

1. 속성 값을 즉시 반환합니다.

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // 속성 식별자
   );
```

2. 함수 실행의 성공 여부에 따라 true 또는 false를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 참조로 전달되는 수신 변수에 배치됩니다.

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,      // 속성 식별자
   double&                        double_var        // 여기서 속성 값을 수락합니다
   );
```

매개 변수

property_id

[in] 포지션 속성 식별자. 이 값은 ENUM_POSITION_PROPERTY_DOUBLE 열거값 중 하나일 수 있습니다.

double_var

[out] 요청된 속성의 값을 수락하는 double 유형의 변수.

반환값

Value of the double type. 이 함수가 실패하면 0이 반환됩니다.

참고

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

포지션에 대한 새로운 데이터를 받으려면 PositionSelect()을 바로 호출한 뒤 참조하는 것이 좋습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 모든 계정 포지션에 의한 루프
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 해당 속성에 액세스할 포지션을 자동으로 선택하여 다음 포지션의 티켓을 가져옵니다.
       ulong ticket=PositionGetTicket(i);
       if(ticket==0)
          continue;

       //--- 포지션 유형을 가져오고 포지션 실제 속성 목록에 대한 헤더를 표시
       string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
       PrintFormat("Double properties of an open position %s #%I64u:", type, ticket);

       //--- 헤더 아래에서 선택한 포지션의 모든 실제 속성을 인쇄
       PositionPropertiesDoublePrint(15);
      }
    /*
    결과:
    Double properties of an open position Buy #2807075208:
    Volume:         1.00
    Price open:     1.10516
    StopLoss:       0.00000
    TakeProfit:     0.00000
    Price current: 1.10518
    Swap:           0.00
    Profit:         2.00 USD
    */
   }
//+------------------------------------------------------------------+
//| 선택한 포지션의 실제 속성을 저널에 표시                                  |
//+------------------------------------------------------------------+
void PositionPropertiesDoublePrint(const uint header_width=0)
   {
    uint   w=0;
    string header="";
    double value=0;

//--- 계정 통화, 포지션 심볼 및 심볼의 소수 자릿수를 가져옵니다.
    string currency=AccountInfoString(ACCOUNT_CURRENCY);
    string symbol  =PositionGetString(POSITION_SYMBOL);
    int    digits  =(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- 헤더 텍스트와 헤더 필드의 너비를 정의합니다.
//--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 10이 됩니다.
    header="Volume:";
    w=(header_width==0 ? header.Length()+1 : header_width);
//--- 저널에서 지정된 헤더 너비로 포지션 볼륨을 가져오고 표시
    if(!PositionGetDouble(POSITION_VOLUME, value))
       return;
    PrintFormat("%-*s%-.2f", w, header, value);

//--- 포지션 가격 값을 저널에 표시
```

© 2000-2025, MetaQuotes Ltd.

```
   header="Price open:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PRICE_OPEN, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 저널에 StopLoss 값을 표시
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 저널에 TakeProfit 값을 표시
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- Price current 값을 저널에 표시
   header="Price current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PRICE_CURRENT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 누적된 스왑값을 저널에 표시
   header="Swap:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_SWAP, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 저널에 현재 수익 값을 표시
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetDouble(POSITION_PROFIT, value))
      return;
   PrintFormat("%-*s%-.2f %s", w, header, value, currency);
  }
```

참고 항목

PositionGetSymbol(), PositionSelect(), Position Properties

© 2000-2025, MetaQuotes Ltd.

PositionGetInteger

이 함수는 PositionGetSymbol 또는 PositionSelect를 사용하여 사전 선택된 오픈 포지션의 요청된 속성을 반환합니다. 포지션 속성은 datetime, int 유형이어야 합니다. 이 함수엔 2가지 변수가 있습니다

1. 속성 값을 즉시 반환합니다.

```
long  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id      // 속성 식별자
   );
```

2. 함수 실행의 성공 여부에 따라 true 또는 false를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 참조로 전달되는 수신 변수에 배치됩니다.

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // 속성 식별자
   long&                           long_var         // 여기서 속성 값을 수락합니다
   );
```

Parameter

property_id

[in] 포지션 속성 식별자. 이 값은 ENUM_POSITION_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

long_var

[out] 요청된 속성 값을 수락하는 long 유형의 변수.

반환값

long 타입의 값. 이 함수가 실패하면 0이 반환됩니다.

참고

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

포지션에 대한 새로운 데이터를 받으려면 PositionSelect()을 바로 호출한 뒤 참조하는 것이 좋습니다.

예:

```
//+------------------------------------------------------------------+
//| 거래 함수                                                        |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- 포지션이 있는지 확인하고 변경 시간을 표시합니다
   if(PositionSelect(_Symbol))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 추가 작업을 위해 포지션 ID를 받습니다
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," position #",position_ID);
//--- 1970년 1월 1일 이후 밀리세컨드 형태의 포지션 시간 수령
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Position #%d  POSITION_TIME_MSC = %i64 milliseconds => %s",positior
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- 1970년 1월 1일 이후 초 단위의 포지션의 최근 변화 시간 수령
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE = %i64 seconds => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- 1970년 1월 1일 이후 밀리세컨드 단위의 포지션의 최근 변경 시간 수령
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Position #%d  POSITION_TIME_UPDATE_MSC = %i64 milliseconds => %s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

참고 항목

PositionGetSymbol(), PositionSelect(), Position Properties

© 2000-2025, MetaQuotes Ltd.

PositionGetString

이 함수는 PositionGetSymbol 또는 PositionSelect를 사용하여 사전 선택된 오픈 포지션의 요청된 속성을 반환합니다. 포지션 속성은 문자열 유형이어야 합니다. 이 함수엔 2가지 변수가 있습니다

1. 속성 값을 즉시 반환합니다.

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // 포지션 식별자
   );
```

2. 함수 실행의 성공 여부에 따라 true 또는 false를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 참조로 전달되는 수신 변수에 배치됩니다.

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,     // 속성 식별자
   string&                        string_var       // 여기서 속성 값을 수락합니다
   );
```

매개 변수

property_id

[in] 포지션 속성 식별자. 이 값은 ENUM_POSITION_PROPERTY_STRING 열거값 중 하나일 수 있습니다.

string_var

[out] 요청된 속성 값을 수락하는 문자열 유형의 변수.

반환값

string 유형의 값. 함수가 실패하면 빈 문자열이 반환됩니다.

참고

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

포지션에 대한 새로운 데이터를 받으려면 PositionSelect()을 바로 호출한 뒤 참조하는 것이 좋습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 모든 계정 포지션에 의한 루프
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 해당 속성에 액세스할 포지션을 자동으로 선택하여 다음 포지션의 티켓을 가져옵니다.
       ulong ticket=PositionGetTicket(i);
       if(ticket==0)
          continue;

       //--- 포지션 유형을 가져오고 포지션 문자열 속성 목록의 헤더를 표시합니다.
       string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
       PrintFormat("String properties of an open position %s #%I64u:", type, ticket);

       //--- 헤더 아래에서 선택한 포지션의 모든 문자열 속성을 인쇄
       PositionPropertiesStringPrint(15);
      }
    /*
    결과:
    String properties of an open position Buy #2810798881:
    Symbol:         EURUSD
    Comment:        Test PositionGetString
    External ID:
    */
  }
//+------------------------------------------------------------------+
//| 저널에서 선택한 포지션의 문자열 속성 표시|                                  |
//+------------------------------------------------------------------+
void PositionPropertiesStringPrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value="";

//--- 헤더 텍스트와 헤더 필드의 너비를 정의합니다.
//--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 10이 됩니다.
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- 저널에 지정된 헤더 너비를 가진 포지션 심볼을 가져오고 표시합니다.
   if(!PositionGetString(POSITION_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 포지션 코멘트를 저널에 표시합니다.
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!PositionGetString(POSITION_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 외부 시스템의 포지션 ID를 저널에 표시
   header="External ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!PositionGetString(POSITION_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
```

참고 항목

PositionGetSymbol(), PositionSelect(), Position Properties

© 2000-2025, MetaQuotes Ltd.

PositionGetTicket

이 함수는 오픈 포지션 목록에 지정된 인덱스를 가진 포지션의 티켓을 반환하고 함수 PositionGetDouble, PositionGetInteger, PositionGetString를 사용하여 작업할 포지션을 자동으로 선택합니다.

```
ulong  PositionGetTicket(
   int  index      // 목록에서의 포지션 번호
   );
```

매개 변수

index

[in] 오픈 포지션 목록에 있는 포지션의 인덱스, 숫자는 0부터 시작합니다.

반환값

포지션의 티켓. 함수가 실패하면 0을 반환합니다.

참고

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

포지션에 대한 새로운 데이터를 받으려면 PositionSelect()을 바로 호출한 뒤 참조하는 것이 좋습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 모든 계정 포지션에 의한 루프
   int total=PositionsTotal();
   for(int i=0; i<total; i++)
     {
      //--- 해당 속성에 액세스할 포지션을 자동으로 선택하여 다음 포지션의 티켓을 가져옵니다.
      ulong ticket=PositionGetTicket(i);
      if(ticket==0)
         continue;

      //--- 포지션 유형을 가져오고 선택한 포지션에 대한 설명을 저널에 표시
      string type=(PositionGetInteger(POSITION_TYPE)==POSITION_TYPE ? "Buy" : "Sell");
      PrintFormat("[%d] Selected position %s #%I64u", i, type, ticket);
     }
   /*
  결과:
   [0] Selected position Sell #2810802718
```

© 2000-2025, MetaQuotes Ltd.

```
   [1] Selected position Buy #2810802919
   */
  }
```

참고 항목

PositionGetSymbol(), PositionSelect(), Position Properties

© 2000-2025, MetaQuotes Ltd.

OrdersTotal

현재 주문 수를 반환합니다.

```
int  OrdersTotal();
```

반환값

int 유형의 값.

참고

현재 보류 주문을 클라이언트 터미널의 "도구 상자"의 "거래"탭에도 표시되는 포지션과 혼동하지 마십시오 주문은 거래를 수행하기 위한 요청이며, 포지션은 하나 이상의 거래의 결과입니다.

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 계정의 활성 펜딩 주문 수를 저널에 가져와 인쇄
   int total=OrdersTotal();
   Print("Number of active pending orders on the account: ", total);
   /*
  결과:
   Number of active pending orders on the account: 2
   */
  }
```

참고 항목

OrderSelect(), OrderGetTicket(), Order Properties

© 2000-2025, MetaQuotes Ltd.

OrderGetTicket

해당 주문의 티켓을 반환하고 함수를 사용하여 추가로 작업할 주문을 자동으로 선택합니다.

```
ulong  OrderGetTicket(
   int  index      // 주문 리스트의 번호
   );
```

Parameter

index

[in] 현재 주문 리스트에 있는 주문 수.

반환값

ulong 타입의 값. 이 함수가 실패하면 0이 반환됩니다.

참고

현재 보류 주문을 클라이언트 터미널의 "도구 상자"의 "거래"탭에도 표시되는 포지션과 혼동하지 마십시오 주문은 거래를 수행하기 위한 요청이며, 포지션은 하나 이상의 거래의 결과입니다.

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

함수 OrderGetTicket()은 주문에 대한 데이터를 프로그램 환경으로 복사하고, OrderGetDouble(), OrderGetInteger(), OrderGetString() 호출을 계속하면 이전에 복사된 데이터가 반환됩니다. 즉, 주문 자체가 더 이상 존재하지 않을 수 있지만(또는 오픈 프라이스, 스탑 로스/이익 실현 수준 또는 만료일 변경) 이 주문의 데이터는 여전히 얻을 수 있습니다. 주문에 대한 새로운 데이터를 수신하려면 주문을 참조하기 전에 OrderGetTicket()으로 요청하는 것이 좋습니다.

예:

```
void OnStart()
  {
//--- 주문 속성에서 값을 반환하기 위한 변수
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- 현재 보류 주문 번호
   uint     total=OrdersTotal();
//--- 루프에서 주문 조사
   for(uint i=0;i<total;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 목록에서 포지션에 대한 주문 티켓을 반환
       if((ticket=OrderGetTicket(i))>0)
         {
          //--- 주문 속성 반환
          open_price    =OrderGetDouble(ORDER_PRICE_OPEN);
          time_setup    =(datetime)OrderGetInteger(ORDER_TIME_SETUP);
          symbol        =OrderGetString(ORDER_SYMBOL);
          order_magic   =OrderGetInteger(ORDER_MAGIC);
          positionID    =OrderGetInteger(ORDER_POSITION_ID);
          initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
          type          =EnumToString(ENUM_ORDER_TYPE(OrderGetInteger(ORDER_TYPE)));
          //--- 주문에 대한 정보를 준비하고 보여줌
          printf("#ticket %d %s %G %s at %G was set up at %s",
                 ticket,                  // order ticket
                 type,                    // type
                 initial_volume,          // placed volume
                 symbol,                  // symbol
                 open_price,              // specified open price
                 TimeToString(time_setup)// 주문 시간
                 );
         }
      }
//---
   }
```

참고 항목

OrdersTotal(), OrderSelect(), OrderGetInteger()

© 2000-2025, MetaQuotes Ltd.

OrderSelect

작업할 주문을 선택합니다. 함수가 성공적으로 완료되었으면 true를 반환합니다. 함수 완료에 실패한 경우 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError()를 호출하십시오.

```
bool  OrderSelect(
   ulong   ticket      // 주문 티켓
   );
```

매개 변수

ticket

[in] 주문 티켓.

반환값

bool 유형의 값.

참고

현재 보류 주문을 클라이언트 터미널의 "도구 상자"의 "거래" 탭에도 표시되는 포지션과 혼동하지 마십시오.

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

함수 OrderSelect()는 주문에 대한 데이터를 프로그램 환경으로 복사하고, 추가로 OrderGetDouble(), OrderGetInteger(), OrderGetString()를 호출하면 이전에 복사된 데이터가 반환됩니다. 즉, 주문 자체가 더 이상 존재하지 않을 수 있지만(또는 오픈 프라이스, 스탑 로스/이익 실현 수준 또는 만료일 변경) 이 주문의 데이터는 여전히 얻을 수 있습니다. 주문에 대한 새로운 데이터를 수신하려면 OrderSelect()을 호출하여 해당 주문을 참조하는 것이 좋습니다.

예:

```
#define   EXPERT_MAGIC  123456
#define   OFFSET        50                    // offset from the current price to pl
#define   DIRECTION     ORDER_TYPE_BUY_LIMIT  // order type
#define   VOLUME        1.0                   // volume
#define   DEVIATION     2                     // allowed deviation from the price

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//-- 거래 요청, 결과 및 변수를 선언하고 초기화
   MqlTradeRequest request={};
   MqlTradeResult  result ={};
```

© 2000-2025, MetaQuotes Ltd.

```
   double          order_price=0;

//--- 펜딩 주문 접수 매개 변수들
   request.action    = TRADE_ACTION_PENDING;                                  // 주문 동작
   request.symbol    = _Symbol;                                               // 심볼
   request.volume    = VOLUME;                                                // 볼륨
   request.deviation = DEVIATION;                                             // 가격으로
   request.magic     = EXPERT_MAGIC;                                          // 주문 매직

//--- 동작 유형 확인
   switch(DIRECTION)
     {
      case ORDER_TYPE_BUY_LIMIT :
        request.type = ORDER_TYPE_BUY_LIMIT;                                  // 주문 유형
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_ASK)-OFFSET*_Point;// 시가
        request.price= NormalizeDouble(order_price, _Digits);                 // 노멀라이즈
        break;
      case ORDER_TYPE_SELL_LIMIT :
        request.type = ORDER_TYPE_SELL_LIMIT;                                 // 주문 유형
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_BID)+OFFSET*_Point;// 시가
        request.price= NormalizeDouble(order_price,_Digits);                  // 노멀라이즈
        break;
      case ORDER_TYPE_BUY_STOP :
        request.type = ORDER_TYPE_BUY_STOP;                                   // 주문 유형
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_ASK)+OFFSET*_Point;// 시가
        request.price= NormalizeDouble(order_price,_Digits);                  // 노멀라이즈
        break;
      case ORDER_TYPE_SELL_STOP :
        request.type = ORDER_TYPE_SELL_STOP;                                  // 주문 유형
        order_price  = SymbolInfoDouble(_Symbol, SYMBOL_BID)-OFFSET*_Point;// 시가
        request.price= NormalizeDouble(order_price,_Digits);                  // 노멀라이즈
        break;
      default: // 만약 펜딩이 아니 주문 혹은 스탑리밋 주문이 선택된 경우
        Alert("This example is only for placing pending orders BuyLimit, SellLimit, Bu
        break;
     }

//--- 요청 전송 요청 전송에 실패한 경우 오류 코드를 표시하고 작업을 완료
   if(!OrderSend(request, result))
     {
      Print("OrderSend error ", GetLastError());
      return;
     }

//--- 작동 데이터 표시
   PrintFormat("Trade request result: retcode=%u, order=%I64u", result.retcode, result

//--- 거래 수행 결과에서 주문 티켓을 가져와 티켓별로 주문을 선택
   ulong ticket=result.order;
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
   if(!OrderSelect(ticket))
     {
      PrintFormat("OrderSelect(%I64u) failed. Error %d", ticket, GetLastError());
      return;
     }

//--- 티켓에서 선택한 주문 데이터를 저널에 표시
   ENUM_ORDER_TYPE   type   = (ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE);
   long              time   = OrderGetInteger(ORDER_TIME_SETUP_MSC);
   double            price = OrderGetDouble(ORDER_PRICE_OPEN);
   double            volume= OrderGetDouble(ORDER_VOLUME_CURRENT);
   string            symbol= OrderGetString(ORDER_SYMBOL);
   int               digits= (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   PrintFormat("Current selected order: %s %.2f %s #%I64u at %.*f, %s",
               symbol, volume, OrderTypeDescription(type), ticket, digits, price, Time
   /*
   결과:
   Trade request result: retcode=10009, order=2811006719
   Current selected order: EURUSD 1.00 Buy Limit #2811006719 at 1.10550, 2024.09.04 1
   */
  }
//+------------------------------------------------------------------+
//| 말리세컨드로 시간을 반환                                          |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                             |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
  }
```

참고 항목

© 2000-2025, MetaQuotes Ltd.

OrderGetInteger(), OrderGetDouble(), OrderGetString(), OrderCalcProfit(), OrderGetTicket(), Order Properties

© 2000–2025, MetaQuotes Ltd.

OrderGetDouble

OrderGetTicket 또는 OrderSelect를 사용하여 미리 선택된 주문의 요청된 속성을 반환합니다. 주문 속성은 double 유형이어야 합니다. 이 함수엔 2가지 변수가 있습니다

1. 속성 값을 즉시 반환합니다.

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 True 또는 False를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 참조로 전달되는 대상 변수에 배치됩니다.

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // 속성 식별자
   double&                     double_var         // 여기서 속성 값을 수락합니다
   );
```

매개 변수

property_id

[in] 주문 속성 식별자. 이 값은 ENUM_ORDER_PROPERTY_DOUBLE 열거값 중 하나일 수 있습니다.

double_var

[out] 요청된 속성 값을 수락하는 double 유형의 변수.

반환값

Value of the double type. 이 함수가 실패하면 0이 반환됩니다.

참고

현재 보류 주문을 클라이언트 터미널의 "도구 상자"의 "거래" 탭에도 표시되는 포지션과 혼동하지 마십시오.

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

주문에 대한 새로운 데이터를 수신하려면, 해당 데이터를 참조하기 전에 OrderSelect()를 요청하는 것이 좋습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 모든 계정 주문 목록에 의한 루프
```

© 2000-2025, MetaQuotes Ltd.

```
int total=OrdersTotal();
for(int i=0; i<total; i++)
  {
   //--- 루프 인덱스를 통해 리스트에서 주문 티켓 가져오기
   ulong ticket=OrderGetTicket(i);
   if(ticket==0)
      continue;

   //--- 주문 유형을 가져오고 선택한 주문의 실제 속성 목록에 대한 헤더를 표시합니다.
   string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
   PrintFormat("Double properties of an active pending order %s #%I64u:", type, tic

   //--- 헤더 아래에서 선택한 주문의 모든 실제 속성을 인쇄합니다.
   OrderPropertiesDoublePrint(16);
  }
/*
결과:
Double properties of an active pending order Sell Limit #2812000714:
Volume initial: 1.00
Volume current: 1.00
Price open:     145.282
StopLoss:       0.000
TakeProfit:     0.000
Price current:  145.044
StopLimit:      0.000
Double properties of an active pending order Buy Limit #2812001112:
Volume initial: 1.00
Volume current: 1.00
Price open:     144.836
StopLoss:       0.000
TakeProfit:     0.000
Price current:  145.051
StopLimit:      0.000
Double properties of an active pending order Buy Stop #2812001488:
Volume initial: 0.50
Volume current: 0.50
Price open:     1.10642
StopLoss:       0.00000
TakeProfit:     0.00000
Price current:  1.10530
StopLimit:      0.00000
Double properties of an active pending order Sell Stop #2812001712:
Volume initial: 0.50
Volume current: 0.50
Price open:     1.10374
StopLoss:       0.00000
TakeProfit:     0.00000
Price current:  1.10525
StopLimit:      0.00000
```

© 2000-2025, MetaQuotes Ltd.

```
   */
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                            |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
  }
//+------------------------------------------------------------------+
//| 선택한 주문의 실제 속성을 저널에 표시                            |
//+------------------------------------------------------------------+
void OrderPropertiesDoublePrint(const uint header_width=0)
  {
//--- 주문 심볼과 심볼의 소수 자릿수를 가져옵니다.
   string symbol = OrderGetString(ORDER_SYMBOL);
   int    digits = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- 저널에 지정된 너비의 헤더로 주문할 때 초기 볼륨을 표시
   OrderPropertyPrint("Volume initial:",header_width,2,ORDER_VOLUME_INITIAL);

//--- 미체결 주문량을 저널에 표시
   OrderPropertyPrint("Volume current:",header_width,2,ORDER_VOLUME_CURRENT);

//--- 주문에서 지정한 가격을 저널에 표시
   OrderPropertyPrint("Price open:",header_width,digits,ORDER_PRICE_OPEN);

//--- 저널에 StopLoss 레벨을 표시
   OrderPropertyPrint("StopLoss:",header_width,digits,ORDER_SL);

//--- 저널에 TakeProfit 레벨을 표시
   OrderPropertyPrint("TakeProfit:",header_width,digits,ORDER_TP);

//--- 저널의 주문 심볼로 현재 가격을 표시
   OrderPropertyPrint("Price current:",header_width,digits,ORDER_PRICE_CURRENT);

//--- StopLimit 주문이 활성화되면 지정가 주문 가격을 저널에 표시
   OrderPropertyPrint("StopLimit:",header_width,digits,ORDER_PRICE_STOPLIMIT);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 주문 실제 자산 가치를 저널에 표시                                |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, int digits, ENUM_ORDE
  {
   double value=0;
   if(!OrderGetDouble(property, value))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
   else
     {
      //--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 1
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-.*f", w, header, digits, value);
     }
  }
```

참고 항목

OrdersTotal(), OrderGetTicket(), Order Properties

© 2000-2025, MetaQuotes Ltd.

OrderGetInteger

OrderGetTicket 또는 OrderSelect를 사용하여 미리 선택된 요청된 주문 속성을 반환합니다.. 주문 속성은 datetime, int 유형이어야 합니다. 이 함수엔 2가지 변수가 있습니다

1. 속성 값을 즉시 반환합니다.

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 true 또는 false를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 레퍼런스로 전달되는 대상 변수에 배치됩니다.

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // 속성 식별자
   long&                        long_var           // 여기서 속성 값을 수락합니다
   );
```

매개 변수

property_id

[in] 주문 속성 식별자. 이 값은 ENUM_ORDER_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

long_var

[out] 요청된 속성 값을 수락하는 long 타입의 변수.

반환값

long 타입의 값. 이 함수가 실패하면 0이 반환됩니다.

참고

현재 보류 주문을 클라이언트 터미널의 "도구 상자"의 "거래" 탭에도 표시되는 포지션과 혼동하지 마십시오.

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

주문에 대한 새로운 데이터를 수신하려면, 해당 데이터를 참조하기 전에 OrderSelect()를 요청하는 것이 좋습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 모든 계정 주문 목록에 의한 루프
```

© 2000-2025, MetaQuotes Ltd.

```
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- 루프 인덱스를 통해 리스트에서 주문 티켓 가져오기
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 주문 유형을 가져오고 선택한 주문의 실제 속성 목록에 대한 헤더를 표시합니다.
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
      PrintFormat("Integer properties of an active pending order %s #%I64u:", type, ti

      //--- 헤더 아래에서 선택한 주문의 모든 정수 속성을 인쇄
      OrderPropertiesIntegerPrint(17);
     }
   /*
   결과:
   Integer properties of an active pending order Buy Limit #2812945317:
   Ticket:           2812945317
   Time setup:       2024.09.04 19:17:16
   Type:             Buy Limit
   State:            Placed
   Time expiration:  0
   Time done:        0
   Time setup msc:   2024.09.04 19:17:16.686
   Time done msc:    0
   Type filling:     Return
   Type time:        Time GTC
   Magic:            0
   Reason:           Client
   Position ID:      0
   Position By ID:   0
   */
  }
//+------------------------------------------------------------------+
//| 저널에서 선택한 주문의 정수 속성 표시                                |
//+------------------------------------------------------------------+
void OrderPropertiesIntegerPrint(const uint header_width=0)
  {
   uint   w=0;
   string header="";
   long   value=0;

//--- 주문 티켓을 저널에 표시
   OrderPropertyPrint("Ticket:", header_width, ORDER_TICKET);

//--- 주문 시간을 저널에 표시
   OrderPropertyPrint("Time setup:", header_width, ORDER_TIME_SETUP);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 주문 유형을 저널에 표시
   OrderPropertyPrint("Type:", header_width, ORDER_TYPE);

//--- 주문 상태를 저널에 표시
   OrderPropertyPrint("State:", header_width, ORDER_STATE);

//--- 주문 유효 기간을 저널에 표시
   OrderPropertyPrint("Time expiration:", header_width, ORDER_TIME_EXPIRATION);

//--- 주문 실행/만료 시간을 저널에 표시
   OrderPropertyPrint("Time done:", header_width, ORDER_TIME_DONE);

//--- 1970년 1월 1일 이후 실행 명령을 내린 시간을 밀리초 단위로 저널에 표시
   OrderPropertyPrint("Time setup msc:", header_width, ORDER_TIME_SETUP_MSC);

//--- 1970년 1월 1일 이후 주문 실행/만료 시간을 밀리초 단위로 저널에 표시합니다.
   OrderPropertyPrint("Time done msc:", header_width, ORDER_TIME_DONE_MSC);

//--- 실행 유형을 저널의 잔여로 표시합니다.
   OrderPropertyPrint("Type filling:", header_width, ORDER_TYPE_FILLING);

//--- 주문 수명을 저널에 표시
   OrderPropertyPrint("Type time:", header_width, ORDER_TYPE_TIME);

//--- 주문을 수행한 EA의 ID를 저널에 표시
   OrderPropertyPrint("Magic:", header_width, ORDER_MAGIC);

//--- 주문 이유 또는 출처를 저널에 표시
   OrderPropertyPrint("Reason:", header_width, ORDER_REASON);

//--- 실행 중 주문에 설정된 포지션 ID를 저널에 표시
   OrderPropertyPrint("Position ID:", header_width, ORDER_POSITION_ID);

//--- 저널에 ORDER_TYPE_CLOSE_BY 유형 주문에 대한 반대 포지션 ID를 표시
   OrderPropertyPrint("Position By ID:", header_width, ORDER_POSITION_BY_ID);
  }
//+------------------------------------------------------------------+
//| 저널에 주문 정수 속성 값을 표시                                  |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, ENUM_ORDER_PROPERTY_IN
  {
   string svalue="";
   long   lvalue=0;
   if(!OrderGetInteger(property, lvalue))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
   else
     {
      switch(property)
        {
```

© 2000-2025, MetaQuotes Ltd.

```
         case ORDER_TICKET          :
         case ORDER_MAGIC           :
         case ORDER_POSITION_ID     :
         case ORDER_POSITION_BY_ID  :
           svalue=(string)lvalue;
           break;

         case ORDER_TIME_SETUP      :
         case ORDER_TIME_EXPIRATION :
         case ORDER_TIME_DONE       :
           svalue=(lvalue!=0 ? TimeToString((datetime)lvalue, TIME_DATE|TIME_MINUTES|T
           break;

         case ORDER_TIME_SETUP_MSC  :
         case ORDER_TIME_DONE_MSC   :
           svalue=(lvalue!=0 ? TimeMscToString(lvalue) : "0");
           break;

         case ORDER_TYPE            :
           svalue=OrderTypeDescription((ENUM_ORDER_TYPE)lvalue);
           break;
         case ORDER_STATE           :
           svalue=OrderStateDescription((ENUM_ORDER_STATE)lvalue);
           break;
         case ORDER_TYPE_FILLING    :
           svalue=OrderTypeFillingDescription((ENUM_ORDER_TYPE_FILLING)lvalue);
           break;
         case ORDER_TYPE_TIME       :
           svalue=OrderTypeTimeDescription((ENUM_ORDER_TYPE_TIME)lvalue);
           break;
         case ORDER_REASON          :
           svalue=OrderReasonDescription((ENUM_ORDER_REASON)lvalue);
           break;

         default                    :
           svalue="Unknown property";
           break;
        }

      //--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 1
      uint w=(header_width==0 ? header.Length()+1 : header_width);
      PrintFormat("%-*s%-s", w, header, svalue);
     }
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                            |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| 주문 상태 설명 반환                                              |
//+------------------------------------------------------------------+
string OrderStateDescription(ENUM_ORDER_STATE state)
  {
   switch(state)
     {
      case ORDER_STATE_STARTED         :  return("Started");
      case ORDER_STATE_PLACED          :  return("Placed");
      case ORDER_STATE_CANCELED        :  return("Canceled");
      case ORDER_STATE_PARTIAL         :  return("Partial");
      case ORDER_STATE_FILLED          :  return("Filled");
      case ORDER_STATE_REJECTED        :  return("Rejected");
      case ORDER_STATE_EXPIRED         :  return("Expired");
      case ORDER_STATE_REQUEST_ADD     :  return("Request Add");
      case ORDER_STATE_REQUEST_MODIFY  :  return("Request Modify");
      case ORDER_STATE_REQUEST_CANCEL  :  return("Request Cancel");
      default                          :  return("Unknown state: "+(string)state);
     }
  }
//+------------------------------------------------------------------+
//| 주문 볼륨 채우기 정책 설명 반환                                  |
//+------------------------------------------------------------------+
string OrderTypeFillingDescription(const ENUM_ORDER_TYPE_FILLING type)
  {
   switch(type)
     {
      case ORDER_FILLING_FOK     :  return("Fill or Kill");
      case ORDER_FILLING_IOC     :  return("Immediate or Cancel");
      case ORDER_FILLING_BOC     :  return("Book or Cancel");
      case ORDER_FILLING_RETURN  :  return("Return");
      default                    :  return("Unknown type filling: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| 주문 만료일 설명 반환                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
string OrderTypeTimeDescription(const ENUM_ORDER_TYPE_TIME type)
  {
   switch(type)
     {
      case ORDER_TIME_GTC           :   return("Time GTC");
      case ORDER_TIME_DAY           :   return("Time Day");
      case ORDER_TIME_SPECIFIED     :   return("Time Specified");
      case ORDER_TIME_SPECIFIED_DAY :   return("Time Specified Day");
      default                       :  return("Unknown type time: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| 주문 접수 이유 설명 반환                                         |
//+------------------------------------------------------------------+
string OrderReasonDescription(const ENUM_ORDER_REASON reason)
  {
   switch(reason)
     {
      case ORDER_REASON_CLIENT    :  return("Client");
      case ORDER_REASON_MOBILE    :  return("Mobile");
      case ORDER_REASON_WEB       :  return("Web");
      case ORDER_REASON_EXPERT    :  return("Expert");
      case ORDER_REASON_SL        :  return("Stop Loss");
      case ORDER_REASON_TP        :  return("Take Profit");
      case ORDER_REASON_SO        :  return("Stop Out");
      default                     :  return("Unknown reason: "+(string)reason);
     }
  }
//+------------------------------------------------------------------+
//| 말리세컨드로 시간을 반환                                         |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 
  }
```

참고 항목

OrdersTotal(), OrderGetTicket(), Order Properties

© 2000-2025, MetaQuotes Ltd.

OrderGetString

OrderGetTicket 또는 OrderSelect를 사용하여 미리 선택된 요청된 주문 속성을 반환합니다.. 주문 속성은 string 유형이어야 합니다. 이 함수엔 2가지 변수가 있습니다

1. 속성 값을 즉시 반환합니다.

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id        // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 True 또는 False를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 레퍼런스로 전달되는 대상 변수에 배치됩니다.

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,       // 속성 식별자
   string&                     string_var         // 여기서 속성 값을 수락합니다
   );
```

매개 변수

property_id

[in] 주문 속성 식별자. 이 값은 ENUM_ORDER_PROPERTY_STRING 열거값 중 하나일 수 있습니다.

string_var

[out] 요청된 속성의 값을 수락하는 string 유형의 변수.

반환값

string 타입의 변수.

참고

현재 보류 주문을 클라이언트 터미널의 "도구 상자"의 "거래" 탭에도 표시되는 포지션과 혼동하지 마십시오.

포지션 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)의 "netting" 해석의 경우, 언제든지 하나의 포지션이 하나의 심볼에 대해 존재할 수 있습니다. 이 포지션은 하나 이상의 거래의 결과입니다. Toolbox 창의 Trading 탭에도 표시되는 유효한 보류 주문과 포지션을 혼동하지 마십시오

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), 하나의 심볼로 여러 포지션을 오픈할 수 있습니다.

주문에 대한 새로운 데이터를 수신하려면, 해당 데이터를 참조하기 전에 OrderSelect()를 요청하는 것이 좋습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 모든 계정 주문 목록에 의한 루프
```

© 2000-2025, MetaQuotes Ltd.

```
   int total=OrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- 루프 인덱스를 통해 리스트에서 주문 티켓 가져오기
      ulong ticket=OrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 주문 유형을 가져오고 선택한 주문의 문자열 속성 목록에 대한 헤더를 표시
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE));
      PrintFormat("String properties of an active pending order %s #%I64u:", type, tic

      //--- 헤더 아래에서 선택한 주문의 모든 문자열 속성을 인쇄
      OrderPropertiesStringPrint(13);
     }
   /*
   결과:
   String properties of an active pending order Sell Limit #2813781342:
   Comment:      Test OrderGetString
   Symbol:       EURUSD
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| 선택한 주문의 문자열 속성을 저널에 표시                                  |
//+------------------------------------------------------------------+
void OrderPropertiesStringPrint(const uint header_width=0)
  {
//--- 코멘트를 저널에 표시
   OrderPropertyPrint("Comment:", header_width, ORDER_COMMENT);

//--- 주문이 접수된 심볼을 저널에 표시
   OrderPropertyPrint("Symbol:", header_width, ORDER_SYMBOL);

//--- 외부 시스템의 주문 ID를 저널에 표시
   OrderPropertyPrint("External ID:", header_width, ORDER_EXTERNAL_ID);
  }
//+------------------------------------------------------------------+
//| 저널에 주문 문자열 속성 값 표시                                       |
//+------------------------------------------------------------------+
void OrderPropertyPrint(const string header, uint header_width, ENUM_ORDER_PROPERTY_ST
  {
   string value="";
   if(!OrderGetString(property, value))
      PrintFormat("Cannot get property %s, error=%d", EnumToString(property), GetLastE
   else
     {
      //--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 1
      uint w=(header_width==0 ? header.Length()+1 : header_width);
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("%-*s%-s", w, header, value);
     }
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                              |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
```

참고 항목

OrdersTotal(), OrderGetTicket(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistorySelect

지정된 기간 동안의 거래 및 주문 기록을 검색합니다.

```
bool  HistorySelect(
   datetime  from_date,     // 시작일
   datetime  to_date        // 종료일
   );
```

Parameter

from_date

[in] 요청 시작일.

to_date

[in] 요청 종료일.

반환값

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

참고

HistorySelect() 는 mql5 프로그램에서 주문 목록과 거래 목록을 만들어 해당 함수를 사용하여 목록 요소를 추가로 참조합니다. 거래 목록의 크기는 HistoryDealsTotal() 함수를 사용하여 반환할 수 있으며, 내역에서의 주문 목록의 크기는 HistoryOrdersTotal()를 사용하여 얻을 수 있습니다. HistoryOrderGetTicket()는 주문 목록의 아이템을 더 잘 선택하므로 HistoryDealGetTicket() 거래 목록의 아이템을 더 잘 선택해야 합니다.

HistoryOrderSelect()를 사용한 후, 티켓별 주문 검색이 성공적으로 완료되면 mql5 프로그램에서 사용할 수 있는 주문 내역 목록이 재설정되고 발견된 주문으로 다시 채워집니다. mql5 프로그램에서 사용할 수 있는 거래 목록에도 동일한 사항이 적용됩니다 – 티켓 번호로 거래를 성공적으로 수신한 경우 HistoryDealSelect()에 의해 재설정되고 다시 채워집니다.

예:

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- 거래 이력 요청
   HistorySelect(0,TimeCurrent());
//--- 개체 생성
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
   long     entry;
//--- 모든 거래에 대해
```

© 2000-2025, MetaQuotes Ltd.

```
   for(uint i=0;i<total;i++)
     {
      //--- 거래 티켓을 얻으려 시도
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- 거래 속성 획득
         price =HistoryDealGetDouble(ticket,DEAL_PRICE);
         time  =(datetime)HistoryDealGetInteger(ticket,DEAL_TIME);
         symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
         type  =HistoryDealGetInteger(ticket,DEAL_TYPE);
         entry =HistoryDealGetInteger(ticket,DEAL_ENTRY);
         profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
         //--- 현재 심볼 전용
         if(price && time && symbol==Symbol())
           {
            //--- 가격 개체 생성
            name="TradeHistory_Deal_"+string(ticket);
            if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
            else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
            //--- 개체 속성 설정
            ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
            ObjectSetInteger(0,name,OBJPROP_BACK,0);
            ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
            if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit
           }
        }
     }
//--- 차트에 적용
   ChartRedraw();
  }
```

참고 항목

HistoryOrderSelect(), HistoryDealSelect()

© 2000-2025, MetaQuotes Ltd.

HistorySelectByPosition

지정된 포지션 식별자가 있는 거래 및 주문 기록을 검색합니다.

```
bool  HistorySelectByPosition(
   ulong   position_id     // 포지션 식별자 - POSITION IDENTIFIER
   );
```

매개 변수

position_id

[in] 실행된 모든 주문 및 모든 딜로 설정된 포지션 식별자.

반환값

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

참고

거래 내역의 주문을 "도구 상자" 표시줄의 "거래" 탭에 표시되는 현재 보류 주문과 혼동하지 마십시오. 취소되었거나 거래로 이어진 주문 목록은 클라이언트 터미널의 "도구 상자"에 있는 "내역" 탭에서 볼 수 있습니다.

HistorySelectByPosition()은 mql5 프로그램에서 주문 목록과 지정된 포지션 식별자의 거래 목록을 만들어 해당 함수를 사용하여 목록의 요소를 추가로 참조합니다. 거래 목록의 크기를 알기 위해 함수 HistoryDealsTotal()를 사용하고, 내역에서의 주문 목록의 크기는 HistoryOrdersTotal()를 사용하여 구할 수 있습니다. 주문 목록의 요소를 살펴보려면, HistoryOrderGetTicket()를 HistoryDealGetTicket()에 대해 사용하십시오.

HistoryOrderSelect()를 사용한 후에는 mql5 프로그램에서 사용할 수 있는 기록 주문 목록이 재설정되고 티켓으로 주문 검색에 성공한 경우 발견된 주문으로 다시 채워집니다. 마찬가지로 mql5 프로그램에서 사용할 수 있는 거래 목록을 참조합니다 – 이 목록은 함수 HistoryDealSelect()에 의해 재설정되며, 티켓 번호에 의해 거래가 성공적으로 검색되면 다시 작성됩니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   long  pos_id_array[];          // 포지션 ID를 저장하기 위한 배열

//--- 전체 히스토리 요청
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- 배열의 펜딩 주문에서만 모든 포지션 ID를 수집
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       ulong ticket=HistoryOrderGetTicket(i);
       if(ticket==0)
          continue;
       ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket, ORDER_TYPE)
       long pos_id=HistoryOrderGetInteger(ticket, ORDER_POSITION_ID);
       if(type<=ORDER_TYPE_SELL || pos_id==0)
          continue;

       int size=ArraySize(pos_id_array);
       if(ArrayResize(pos_id_array, size+1)==size+1)
          pos_id_array[size]=pos_id;
      }

//--- 배열의 포지션 ID의 리스트로
   total=ArraySize(pos_id_array);
   for(int i=0; i<total; i++)
     {
      //--- 헤더와 포지션 주문 및 거래 리스트를 인쇄
      long position_id=pos_id_array[i];
      Print("List of orders and deals for position with ID: ", position_id);
      HistorySelectByPositionProcess(position_id);
     }
   /*
   결과:
   List of orders and deals for position with ID: 1819629924
     [0] Order Sell Limit #1819629924
     [1] Order Buy #1819633194
     [0] Entry In Deal Sell #1794972472
     [1] Entry Out Deal Buy #1794975589
   List of orders and deals for position with ID: 1841753970
     [0] Order Sell Stop #1841753970
     [1] Order Buy #1842322160
     [0] Entry In Deal Sell #1817242142
     [1] Entry Out Deal Buy #1817765341
   */
  }
//+------------------------------------------------------------------+
//| 포지션 ID별로 주문 및 거래 내역을 선택하고                          |
//| 저널의 해당 위치에 대한 주문 및 거래 목록을 인쇄|                   |
//+------------------------------------------------------------------+
bool HistorySelectByPositionProcess(const long position_id)
  {
//--- 지정된 포지션 ID를 가진 거래 및 주문 내역을 요청
   if(!HistorySelectByPosition(position_id))
     {
      PrintFormat("HistorySelectByPosition(%I64d) failed. Error %d", position_id, GetI
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 포지션 주문 리스트를 인쇄
   int orders_total=HistoryOrdersTotal();
   for(int i=0; i<orders_total; i++)
     {
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_ORDER_TYPE order_type=(ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket, ORDE
      PrintFormat("  [%d] Order %s #%I64u", i, OrderTypeDescription(order_type), ticke
     }

//--- 저널의 포지션 거래 리스트를 인쇄
   int deals_total =HistoryDealsTotal();
   for(int i=0; i<deals_total; i++)
     {
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;
      ENUM_DEAL_ENTRY deal_entry=(ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket, DEAL_E
      ENUM_DEAL_TYPE  deal_type= (ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DEAL_TY
      if(deal_type!=DEAL_TYPE_BUY && deal_type!=DEAL_TYPE_SELL)
         continue;
      PrintFormat("  [%d] Entry %s Deal %s #%I64u", i, DealEntryDescription(deal_entry
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                                |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| 포지션 거래 유형 설명을 반환                                          |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   switch(type)
     {
      //--- 매수 및 매도 거래에 대한 설명만 반환
      //--- 다른 모든 유형은 해당 포지션에 적용되지 않으므로
      case DEAL_TYPE_BUY    :  return("Buy");
      case DEAL_TYPE_SELL   :  return("Sell");
      default               :  return("Unknown deal type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| 포지션 변경 메서드 반환                                          |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN       :  return("In");
      case DEAL_ENTRY_OUT      :  return("Out");
      case DEAL_ENTRY_INOUT    :  return("InOut");
      case DEAL_ENTRY_OUT_BY   :  return("Out by");
      case DEAL_ENTRY_STATE    :  return("Status record");
      default                  :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

참고 항목

HistorySelect(), HistoryOrderGetTicket(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistoryOrderSelect

기록에서 적절한 함수를 통해 추가로 호출할 주문을 선택합니다. 함수가 성공적으로 완료되면 true가 반환됩니다. 함수가 실패하면 false가 반환됩니다. 에러에 대한 더 많은 사항에 대해서는 GetLastError()을 호출하십시오.

```
bool  HistoryOrderSelect(
   ulong  ticket      // 주문 티켓
   );
```

매개 변수

ticket

[in] 주문 티켓.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

거래 내역의 주문을 "도구 상자" 표시줄의 "거래" 탭에 표시되는 현재 보류 주문과 혼동하지 마십시오. 취소되었거나 거래로 이어진 주문 목록은 클라이언트 터미널의 "도구 상자"에 있는 "내역" 탭에서 볼 수 있습니다.

HistoryOrderSelect()의 실행이 성공적으로 완료된 경우 내역에서 주문 목록을 지우고 호출에 사용할 수 있으며 단일 주문으로 복사합니다. HistorySelect()에서 선택한 모든 거래를 수행해야 하는 경우, HistoryOrderGetTicket()를 사용하는 것이 좋습니다.

예:

```
#define   TICKET    1819621374   // 알려진 주문 티켓, 예: 계정 히스토리에서
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- TICKET에 지정된 티켓으로 과거 주문을 선택
   if(!HistoryOrderSelect(TICKET))
     {
      PrintFormat("HistoryOrderSelect(%I64u) failed. Error %d", TICKET, GetLastError()
      return;
     }

//--- 주문이 성공적으로 선택되면 해당 데이터를 가져오고 저널에 주문 설명을 표시
   ENUM_ORDER_TYPE   order_type  = (ENUM_ORDER_TYPE)HistoryOrderGetInteger(TICKET, ORI
   ENUM_ORDER_STATE  order_state = (ENUM_ORDER_STATE)HistoryOrderGetInteger(TICKET, OR
   ENUM_ORDER_REASON order_reason= (ENUM_ORDER_REASON)HistoryOrderGetInteger(TICKET, O
   long              order_time  = HistoryOrderGetInteger(TICKET, ORDER_TIME_SETUP_MSC
   string            order_symbol=HistoryOrderGetString(TICKET, ORDER_SYMBOL);
   double            order_vol_init=HistoryOrderGetDouble(TICKET, ORDER_VOLUME_INITIAL
   double            order_vol_curr=HistoryOrderGetDouble(TICKET, ORDER_VOLUME_CURRENT
```

© 2000-2025, MetaQuotes Ltd.

```
   PrintFormat("%s Order %.2f/%s %s #%I64u %s by %s at %s",
               order_symbol, order_vol_init, (order_vol_curr>0 ? DoubleToString(order_
               OrderTypeDescription(order_type), TICKET, OrderStateDescription(order_s
               OrderReasonDescription(order_reason), TimeMscToString(order_time));
   /*
   다양한 지정된 티켓에 대한 결과:
   EURUSD Order 0.50/0.50 Buy Limit #2812894647 Canceled by Client at 2024.09.04 19:02
   EURUSD Order 0.10/0 Sell #1753011743 Filled by Take Profit at 2023.06.12 17:04:20.1
   GBPUSD Order 0.10/0 Buy #1819621374 Filled by Client at 2023.07.24 06:16:25.746
   */
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                                |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
//+------------------------------------------------------------------+
//| 주문 상태 설명 반환                                                 |
//+------------------------------------------------------------------+
string OrderStateDescription(ENUM_ORDER_STATE state)
  {
   switch(state)
     {
      case ORDER_STATE_STARTED         :  return("Started");
      case ORDER_STATE_PLACED          :  return("Placed");
      case ORDER_STATE_CANCELED        :  return("Canceled");
      case ORDER_STATE_PARTIAL         :  return("Partial");
      case ORDER_STATE_FILLED          :  return("Filled");
      case ORDER_STATE_REJECTED        :  return("Rejected");
      case ORDER_STATE_EXPIRED         :  return("Expired");
      case ORDER_STATE_REQUEST_ADD     :  return("Request Add");
      case ORDER_STATE_REQUEST_MODIFY  :  return("Request Modify");
      case ORDER_STATE_REQUEST_CANCEL  :  return("Request Cancel");
      default                          :  return("Unknown state: "+(string)state);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 주문 접수 이유 설명 반환                                         |
//+------------------------------------------------------------------+
string OrderReasonDescription(const ENUM_ORDER_REASON reason)
  {
   switch(reason)
     {
      case ORDER_REASON_CLIENT    :  return("Client");
      case ORDER_REASON_MOBILE    :  return("Mobile");
      case ORDER_REASON_WEB       :  return("Web");
      case ORDER_REASON_EXPERT    :  return("Expert");
      case ORDER_REASON_SL        :  return("Stop Loss");
      case ORDER_REASON_TP        :  return("Take Profit");
      case ORDER_REASON_SO        :  return("Stop Out");
      default                     :  return("Unknown reason: "+(string)reason);
     }
  }
//+------------------------------------------------------------------+
//| 말리세컨드로 시간을 반환                                         |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 
  }
```

참고 항목

HistorySelect(), HistoryOrderGetTicket(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistoryOrdersTotal

내역의 주문 수를 반환합니다. HistoryOrdersTotal() 호출 이전에, 먼저 HistorySelect() 또는 HistorySelectByPosition() 함수를 사용하여 거래 및 주문 내역을 수신해야 합니다.

```
int  HistoryOrdersTotal();
```

반환값

int 유형의 값.

참고

거래 내역의 주문을 "도구 상자" 표시줄의 "거래" 탭에 표시되는 현재 보류 주문과 혼동하지 마십시오. 취소되었거나 거래로 이어진 주문 목록은 클라이언트 터미널의 "도구 상자"에 있는 "내역" 탭에서 볼 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 계정의 모든 기존 내역을 요청
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- 목록의 주문 수를 가져와서 저널에 표시
   int total=HistoryOrdersTotal();
   Print("Number of historical orders on the account: ", total);
   /*
   결과:
   Number of historical orders on the account: 496
   */
  }
```

참고 항목

HistorySelect(), HistoryOrderSelect(), HistoryOrderGetTicket(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetTicket

내역에서 해당 주문의 티켓을 반환합니다. HistoryOrderGetTicket() 호출 이전에, 먼저 HistorySelect() 또는 HistorySelectByPosition() 함수를 사용하여 거래 및 주문 내역을 수신해야 합니다.

```
ulong  HistoryOrderGetTicket(
   int  index      // 주문 리스트의 번호
   );
```

Parameter

index

[in] 주문 목록의 주문 번호.

반환값

ulong 타입의 값. 이 함수가 실패하면 0이 반환됩니다.

참고

거래 내역의 주문을 "도구 상자" 표시줄의 "거래" 탭에 표시되는 현재 보류 주문과 혼동하지 마십시오. 취소되었거나 거래로 이어진 주문 목록은 클라이언트 터미널의 "도구 상자"에 있는 "내역" 탭에서 볼 수 있습니다.

예:

```
void OnStart()
  {
   datetime from=0;
   datetime to=TimeCurrent();
//--- 전체 내역 요청
   HistorySelect(from,to);
//--- 주문 속성에서 값을 반환하기 위한 변수
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   datetime time_done;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- 현재 보류 주문 번호
   uint     total=HistoryOrdersTotal();
//--- 루프에서 주문 조사
   for(uint i=0;i<total;i++)
     {
      //--- 목록에서 포지션에 대한 주문 티켓을 반환
      if((ticket=HistoryOrderGetTicket(i))>0)
        {
         //--- 주문 속성 반환
         open_price    =HistoryOrderGetDouble(ticket,ORDER_PRICE_OPEN);
```

© 2000-2025, MetaQuotes Ltd.

```
         time_setup    =(datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_SETUP);
         time_done     =(datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_DONE);
         symbol        =HistoryOrderGetString(ticket,ORDER_SYMBOL);
         order_magic   =HistoryOrderGetInteger(ticket,ORDER_MAGIC);
         positionID    =HistoryOrderGetInteger(ticket,ORDER_POSITION_ID);
         initial_volume=HistoryOrderGetDouble(ticket,ORDER_VOLUME_INITIAL);
         type          =GetOrderType(HistoryOrderGetInteger(ticket,ORDER_TYPE));
         //--- 주문에 대한 정보를 준비하고 보여줌
         printf("#ticket %d %s %G %s at %G was set up at %s => done at %s, pos ID=%d",
                ticket,                  // 주문 티켓
                type,                    // 유형
                initial_volume,          // 주문량
                symbol,                  // 심볼
                open_price,              // 명시된 시가
                TimeToString(time_setup),// 주문 시점
                TimeToString(time_done), // 주문 실행 또는 삭제 시간
                positionID               // 주문 거래가 포함된 포지션의 ID
                );
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| 주문 유형의 문자열 이름을 반환                                   |
//+------------------------------------------------------------------+
string GetOrderType(long type)
  {
   string str_type="unknown operation";
   switch(type)
     {
      case (ORDER_TYPE_BUY):             return("buy");
      case (ORDER_TYPE_SELL):            return("sell");
      case (ORDER_TYPE_BUY_LIMIT):       return("buy limit");
      case (ORDER_TYPE_SELL_LIMIT):      return("sell limit");
      case (ORDER_TYPE_BUY_STOP):        return("buy stop");
      case (ORDER_TYPE_SELL_STOP):       return("sell stop");
      case (ORDER_TYPE_BUY_STOP_LIMIT): return("buy stop limit");
      case (ORDER_TYPE_SELL_STOP_LIMIT):return("sell stop limit");
     }
   return(str_type);
  }
```

참고 항목

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetDouble

요청된 주문 속성을 반환합니다. 주문 속성은 double 유형이어야 합니다. 이 함수의 변수는 두 가지가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
double  HistoryOrderGetDouble(
   ulong                      ticket_number,     // 티켓
   ENUM_ORDER_PROPERTY_DOUBLE  property_id       // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 True 또는 False를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 레퍼런스로 전달되는 대상 변수에 배치됩니다.

```
bool  HistoryOrderGetDouble(
   ulong                      ticket_number,     // 티켓
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,      // 속성 식별자
   double&                    double_var         // 여기서 속성 값을 수락합니다
   );
```

매개 변수

ticket_number

[in] 주문 티켓.

property_id

[in] 주문 속성 식별자. 이 값은 ENUM_ORDER_PROPERTY_DOUBLE 열거값 중 하나일 수 있습니다.

double_var

[out] 요청된 속성 값을 수락하는 double 유형의 변수.

반환값

double 타입의 값.

참고

거래 내역의 주문을 "도구 상자" 표시줄의 "거래" 탭에 표시되는 현재 보류 주문과 혼동하지 마십시오. 취소되었거나 거래로 이어진 주문 목록은 클라이언트 터미널의 "도구 상자"에 있는 "내역" 탭에서 볼 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래와 주문 히스토리 요청
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- 계정의 모든 과거 주문 목록에 의한 루프
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- 루프 인덱스를 통해 리스트에서 주문 티켓 가져오기
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 주문 유형을 가져오고 선택한 주문의 실제 속성 목록에 대한 헤더를 표시합니다.
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket,
      PrintFormat("Double properties of an history order %s #%I64u:", type, ticket);

      //--- 헤더 아래에서 선택한 주문의 모든 실제 속성을 인쇄합니다.
      HistoryOrderPropertiesDoublePrint(ticket, 16);
     }
   /*
   결과:
   Double properties of an history order Sell #2810847541:
   Volume initial: 0.50
   Volume current: 0.00
   Price open:     1.10491
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10491
   StopLimit:      0.00000
   Double properties of an history order Buy Limit #2811003507:
   Volume initial: 1.00
   Volume current: 1.00
   Price open:     1.10547
   StopLoss:       0.00000
   TakeProfit:     0.00000
   Price current:  1.10591
   StopLimit:      0.00000
   */
  }
//+------------------------------------------------------------------+
//| 저널에서 선택한 주문 히스토리의 실제 속성을 표시                          |
//+------------------------------------------------------------------+
void HistoryOrderPropertiesDoublePrint(const long ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   double value=0;

//--- 주문 심볼과 심볼의 소수 자릿수를 가져옵니다.
```

© 2000-2025, MetaQuotes Ltd.

```
   string symbol = HistoryOrderGetString(ticket, ORDER_SYMBOL);
   int    digits = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- 헤더 텍스트와 헤더 필드의 너비를 정의합니다.
//--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 10이 됩니다.
   header="Volume initial:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- 저널에서 지정된 너비의 헤더로 주문할 때 초기 볼륨을 가져오고 표시
   if(!HistoryOrderGetDouble(ticket, ORDER_VOLUME_INITIAL, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 미체결 주문량을 저널에 표시
   header="Volume current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_VOLUME_CURRENT, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- 주문에서 지정한 가격을 저널에 표시
   header="Price open:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_OPEN, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 저널에 StopLoss 레벨을 표시
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 저널에 TakeProfit 레벨을 표시
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 저널의 주문 심볼로 현재 가격을 표시
   header="Price current:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_CURRENT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- StopLimit 주문이 활성화되면 지정가 주문 가격을 저널에 표시
   header="StopLimit:";
```

© 2000-2025, MetaQuotes Ltd.

```
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetDouble(ticket, ORDER_PRICE_STOPLIMIT, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                            |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type");
     }
  }
```

참고 항목

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetInteger

주문의 요청된 속성을 반환합니다. 주문 속성은 datetime, int 유형이어야 합니다. 이 함수의 변수는 두 가지가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
long  HistoryOrderGetInteger(
   ulong                        ticket_number,     // 티켓
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 True 또는 False를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 레퍼런스로 전달되는 대상 변수에 배치됩니다.

```
bool  HistoryOrderGetInteger(
   ulong                        ticket_number,     // 티켓
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // 속성 식별자
   long&                        long_var           // 여기서 속성 값을 수락합니다
   );
```

Parameter

ticket_number

[in] 주문 티켓.

property_id

[in] 주문 속성 식별자. 이 값은 ENUM_ORDER_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

long_var

[out] 요청된 속성 값을 수락하는 long 타입의 변수.

반환값

long 타입 값.

참고

거래 내역의 주문을 "도구 상자" 표시줄의 "거래" 탭에 표시되는 현재 보류 주문과 혼동하지 마십시오. 취소되었거나 거래로 이어진 주문 목록은 클라이언트 터미널의 "도구 상자"에 있는 "내역" 탭에서 볼 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 거래 함수                                                        |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- 주별 거래 이력에서 지난 주문의 티켓 수령
   ulong last_order=GetLastOrderTicket();
   if(HistoryOrderSelect(last_order))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 1970년 1월 1일 이후 주문 시간(밀리초)
      long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
      PrintFormat("Order #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                  last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
      //--- 1970년 1월 1일 이후 주문 실행/취소 시간(밀리초)
      long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
      PrintFormat("Order #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                  last_order,time_done_msc,TimeToString(time_done_msc/1000));
     }
   else // 실패 통지
      PrintFormat("HistoryOrderSelect() failed for #%d. Eror code=%d",
                  last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------+
//| 내역의 마지막 주문 티켓 또는 -1을 반환합니다.                          |
//+------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
//--- 지난 7일 동안의 내역을 요청합니다
   if(!GetTradeHistory(7))
     {
      //--- 요청 실패에 대해 알림 후 -1 반환
      Print(__FUNCTION__," HistorySelect() returned false");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- 주문이 있다면 작업 수행
   if(orders>0)
     {
      Print("Orders = ",orders);
      first_order=HistoryOrderGetTicket(0);
      PrintFormat("first_order = %d",first_order);
      if(orders>1)
        {
         last_order=HistoryOrderGetTicket((int)orders-1);
         PrintFormat("last_order = %d",last_order);
         return last_order;
        }
      return first_order;
     }
//--- 주문이 발견되지 않는다면 -1을 반환합니다
   return -1;
  }
//+------------------------------------------------------------------------+
//| 최근 일들 동안의 내역을 요청하고 장애가 발생한 경우 Fal se를 반환합니다          |
//+------------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool GetTradeHistory(int days)
  {
//--- 거래 내역을 의뢰할 주간의 기간을 설정
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- 요청을 하고 결과 확인
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- 이력 수령 성공
   return true;
  }
```

참고 항목

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistoryOrderGetString

주문의 요청된 속성을 반환합니다. 주문 속성은 string 유형이어야 합니다. 이 함수의 변수는 두 가지가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
string  HistoryOrderGetString(
   ulong                      ticket_number,     // 티켓
   ENUM_ORDER_PROPERTY_STRING  property_id       // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 True 또는 False를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 레퍼런스로 전달되는 대상 변수에 배치됩니다.

```
bool  HistoryOrderGetString(
   ulong                      ticket_number,     // 티켓
   ENUM_ORDER_PROPERTY_STRING  property_id,      // 속성 식별자
   string&                    string_var         // 여기서 속성 값을 수락합니다
   );
```

매개 변수

ticket_number

[in] 주문 티켓.

property_id

[in] 주문 속성 식별자. 이 값은 ENUM_ORDER_PROPERTY_STRING 열거값 중 하나일 수 있습니다.

string_var

[out] string 유형의 변수.

반환값

string 타입의 변수.

참고

거래 내역의 주문을 "도구 상자" 표시줄의 "거래" 탭에 표시되는 현재 보류 주문과 혼동하지 마십시오. 취소되었거나 거래로 이어진 주문 목록은 클라이언트 터미널의 "도구 상자"에 있는 "내역" 탭에서 볼 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래와 주문 히스토리 요청
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- 계정의 모든 과거 주문 목록에 의한 루프
   int total=HistoryOrdersTotal();
   for(int i=0; i<total; i++)
     {
      //--- 루프 인덱스를 통해 리스트에서 주문 티켓 가져오기
      ulong ticket=HistoryOrderGetTicket(i);
      if(ticket==0)
         continue;

      //--- 주문 유형을 가져오고 선택한 주문의 문자열 속성 목록에 대한 헤더를 표시
      string type=OrderTypeDescription((ENUM_ORDER_TYPE)HistoryOrderGetInteger(ticket,
      PrintFormat("String properties of an history order %s #%I64u:", type, ticket);

      //--- 헤더 아래에서 선택한 주문의 모든 문자열 속성을 인쇄
      HistoryOrderPropertiesStringPrint(ticket, 16);
     }
   /*
   결과:
   String properties of an history order Buy #2646074112:
   Comment:         [tp 1.09137]
   Symbol:          EURUSD
   External ID:
   String properties of an history order Buy #2646131906:
   Comment:
   Symbol:          EURUSD
   External ID:
   */
  }
//+------------------------------------------------------------------+
//| 저널에서 선택한 주문 히스토리의 문자열 속성을 표시                      |
//+------------------------------------------------------------------+
void HistoryOrderPropertiesStringPrint(const long ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value="";

//--- 헤더 텍스트와 헤더 필드의 너비를 정의합니다.
//--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 10이 됩니다.
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- 저널에 지정된 헤더 너비의 코멘트를 가져오고 표시
   if(!HistoryOrderGetString(ticket, ORDER_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 주문이 접수된 심볼을 저널에 표시
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetString(ticket, ORDER_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 외부 시스템의 주문 ID를 저널에 표시
   header="External ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryOrderGetString(ticket, ORDER_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
//+------------------------------------------------------------------+
//| 주문 유형 설명을 반환                                            |
//+------------------------------------------------------------------+
string OrderTypeDescription(const ENUM_ORDER_TYPE type)
  {
   switch(type)
     {
      case ORDER_TYPE_BUY              :  return("Buy");
      case ORDER_TYPE_SELL             :  return("Sell");
      case ORDER_TYPE_BUY_LIMIT        :  return("Buy Limit");
      case ORDER_TYPE_SELL_LIMIT       :  return("Sell Limit");
      case ORDER_TYPE_BUY_STOP         :  return("Buy Stop");
      case ORDER_TYPE_SELL_STOP        :  return("Sell Stop");
      case ORDER_TYPE_BUY_STOP_LIMIT   :  return("Buy Stop Limit");
      case ORDER_TYPE_SELL_STOP_LIMIT  :  return("Sell Stop Limit");
      default                          :  return("Unknown order type: "+(string)type);
     }
  }
```

참고 항목

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Order Properties

© 2000-2025, MetaQuotes Ltd.

HistoryDealSelect

내역에서 적절한 함수를 통해 추가로 호출할 거래를 선택합니다. 함수가 성공적으로 완료되면 true가 반환됩니다. 함수가 실패하면 false가 반환됩니다. 에러에 대한 더 많은 사항에 대해서는 GetLastError()을 호출하십시오.

```
bool  HistoryDealSelect(
   ulong  ticket      // 거래 티켓
   );
```

매개 변수

ticket

[in] 거래 티켓.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

주문, 딜 및 포지션을 혼동하지 마십시오. 각 거래는 주문 실행의 결과이고 각 포지션은 하나 이상의 거래의 요약 결과입니다.

HistoryDealSelect()는 참조할 수 있는 거래 목록을 mql5-program에서 지우고 HistoryDealSelect()의 실행이 성공적으로 완료된 경우 단일 거래를 복제합니다. HistorySelect() 함수로 선택한 모든 거래를 수행해야 하는 경우 HistoryDealGetTicket()를 사용하는 것이 좋습니다.

예:

```
#define   TICKET    2620919264   // 예를 들어 터미널 계정 히스토리에서 가져온 알려진 거래의 

long      ExtTicket=TICKET;      // 지정된 티켓을 스크립트의 테스트에 대한 매크로 대체 변수로
                                 // 또는 EA의 OnTradeTransaction() 핸들러에서 거래를 처리
//+------------------------------------------------------------------+
//| Expert TradeTransaction handler                                  |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction& trans,
                        const MqlTradeRequest& request,
                        const MqlTradeResult& result)
  {
   //--- 거래가 히스토리에 특정 거래를 추가하는 경우
   if(trans.type==TRADE_TRANSACTION_DEAL_ADD)
     {
      //--- 티켓으로 거래를 선택하고 해당 데이터를 가져와서 저널에 거래 설명을 표시합니다.
      HistoryDealSelectProcess(trans.deal);
     }
  }

//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
//--- 티켓으로 거래를 선택하고 해당 데이터를 가져오고 저널에 거래 설명을 표시합니다.
   HistoryDealSelectProcess(ExtTicket);
   /*
  결과:
   (Position ID #2645974677) EURUSD Deal Out 0.10 Buy #2620919264 by order #264602896
   */
  }
//+------------------------------------------------------------------+
//| 티켓별로 거래을 선택하고 저널에 거래 데이터를 인쇄합니다.                |
//+------------------------------------------------------------------+
void HistoryDealSelectProcess(const ulong deal_ticket)
  {
//--- deal_ticket에 지정된 티켓으로 과거 거래를 선택합니다.
   ResetLastError();
   if(!HistoryDealSelect(deal_ticket))
     {
      PrintFormat("HistoryDealSelect(%I64u) failed. Error %d", deal_ticket, GetLastErr
      return;
     }

//--- 거래가 성공적으로 선택되면 해당 데이터를 가져오고 저널에 거래 설명을 표시합니다.
   ENUM_DEAL_TYPE    deal_type  = (ENUM_DEAL_TYPE)HistoryDealGetInteger(ExtTicket, DEA
   ENUM_DEAL_ENTRY   deal_entry = (ENUM_DEAL_ENTRY)HistoryDealGetInteger(ExtTicket, DE
   ENUM_DEAL_REASON  deal_reason= (ENUM_DEAL_REASON)HistoryDealGetInteger(ExtTicket, I
   long              deal_time  = HistoryDealGetInteger(ExtTicket, DEAL_TIME_MSC);
   long              deal_order = HistoryDealGetInteger(ExtTicket, DEAL_ORDER);
   long              deal_pos_id= HistoryDealGetInteger(ExtTicket, DEAL_POSITION_ID);
   string            deal_symbol= HistoryDealGetString(ExtTicket, DEAL_SYMBOL);
   double            deal_volume= HistoryDealGetDouble(ExtTicket, DEAL_VOLUME);
   double            deal_price = HistoryDealGetDouble(ExtTicket, DEAL_PRICE);
   int               digits     = (int)SymbolInfoInteger(deal_symbol, SYMBOL_DIGITS);

   PrintFormat("(Position ID #%I64d) %s Deal %s %.2f %s #%I64u by order #%I64d at %.*
               deal_pos_id, deal_symbol, DealEntryDescription(deal_entry), deal_volume
               DealTypeDescription(deal_type), ExtTicket, deal_order, digits, deal_pri
               TimeMscToString(deal_time));
  }
//+------------------------------------------------------------------+
//| 거래 타입 설명을 반환                                              |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                    :  return("Buy");
      case DEAL_TYPE_SELL                   :  return("Sell");
      case DEAL_TYPE_BALANCE                :  return("Balance");
```

© 2000-2025, MetaQuotes Ltd.

```
      case DEAL_TYPE_CREDIT                   :  return("Credit");
      case DEAL_TYPE_CHARGE                   :  return("Additional charge");
      case DEAL_TYPE_CORRECTION               :  return("Correction");
      case DEAL_TYPE_BONUS                    :  return("Bonus");
      case DEAL_TYPE_COMMISSION               :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY         :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY       :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY   :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY:  return("Monthly agent commission");
      case DEAL_TYPE_INTEREST                 :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED             :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED            :  return("Canceled sell deal");
      case DEAL_DIVIDEND                      :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED              :  return("Franked (non-taxable) dividenc
      case DEAL_TAX                           :  return("Tax charges");
      default                                 :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| 포지션 변경 메서드 반환                                          |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
//+------------------------------------------------------------------+
//| 말리세컨드로 시간을 반환                                         |
//+------------------------------------------------------------------+
string TimeMscToString(const long time_msc, int flags=TIME_DATE|TIME_MINUTES|TIME_SEC
  {
   return(TimeToString(time_msc/1000, flags) + "." + IntegerToString(time_msc %1000, 
  }
```

참고 항목

HistorySelect(), HistoryDealGetTicket(), Deal Properties

© 2000-2025, MetaQuotes Ltd.

HistoryDealsTotal

내역에 포함된 거래 수를 반환합니다. HistoryDealsTotal() 호출 전에, 먼저 HistorySelect() 또는 HistorySelectByPosition() 함수를 사용하여 거래 및 주문 내역을 수신해야 합니다.

```
int  HistoryDealsTotal();
```

반환값

int 유형의 값.

참고

주문, 딜 및 포지션을 혼동하지 마십시오. 각 거래는 주문 실행의 결과이고 각 포지션은 하나 이상의 거래의 요약 결과입니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 계정의 모든 기존 내역을 요청
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
     }

//--- 리스트에서 거래 수를 가져와 저널에 표시합니다.
   int total=HistoryDealsTotal();
   Print("Number of historical deals on the account: ", total);
   /*
  결과:
   Number of historical deals on the account: 339
   */
  }
```

참고 항목

HistorySelect(), HistoryDealGetTicket(), Deal Properties

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetTicket

이 함수는 추가적 처리를 위해 거래를 선택하고 거래 티켓을 내역에 반환합니다. HistoryDealGetTicket() 요청 이전에, 먼저 HistorySelect() 또는 HistorySelectByPosition() 함수를 사용하여 거래 및 주문 내역을 수신해야 합니다.

```
ulong  HistoryDealGetTicket(
   int  index      // 티켓 딜
   );
```

Parameter

index

[in] 거래 리스트의 거래 수

반환값

ulong 타입의 값. 이 함수가 실패하면 0이 반환됩니다.

참고

주문, 딜 및 포지션을 혼동하지 마십시오. 각 거래는 주문 실행의 결과이고 각 포지션은 하나 이상의 거래의 요약 결과입니다.

예:

```
void OnStart()
  {
   ulong deal_ticket;            // 딜 티켓
   ulong order_ticket;           // 딜이 실행된 주문의 티켓
   datetime transaction_time;    // 딜이 실행된 시간
   long deal_type ;              // 트레이드 작업의 종류
   long position_ID;             // 포지션 ID
   string deal_description;      // 작업 설명
   double volume;                // 작업량
   string symbol;                // 딜의 심볼
//--- 거래 내역을 요청할 시작 및 종료 날짜를 설정합니다
   datetime from_date=0;         // 최초부터
   datetime to_date=TimeCurrent();// 현재까지
//--- 지정된 기간의 딜 내역을 요청
   HistorySelect(from_date,to_date);
//--- 거래 리스트의 총수
   int deals=HistoryDealsTotal();
//--- 여기에서 각 거래를 처리
   for(int i=0;i<deals;i++)
     {
      deal_ticket=               HistoryDealGetTicket(i);
      volume=                    HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
      transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
      order_ticket=              HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
      deal_type=                 HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
      symbol=                    HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
```

© 2000-2025, MetaQuotes Ltd.

```
      position_ID=              HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
      deal_description=         GetDealDescription(deal_type,volume,symbol,order_tic
      //--- 딜 번호 형식을 잘 갖춥니다
      string print_index=StringFormat("% 3d",i);
      //--- 딜의 정보 보이기
      Print(print_index+": deal #",deal_ticket," at ",transaction_time,deal_descriptio
     }
  }
//+------------------------------------------------------------------+
//| 작업에 대한 문자열 설명을 반환                                   |
//+------------------------------------------------------------------+
string GetDealDescription(long deal_type,double volume,string symbol,long ticket,long
  {
   string descr;
//---
   switch(deal_type)
     {
      case DEAL_TYPE_BALANCE:                  return ("잔고");
      case DEAL_TYPE_CREDIT:                   return ("credit");
      case DEAL_TYPE_CHARGE:                   return ("charge");
      case DEAL_TYPE_CORRECTION:               return ("correction");
      case DEAL_TYPE_BUY:                      descr="buy"; break;
      case DEAL_TYPE_SELL:                     descr="sell"; break;
      case DEAL_TYPE_BONUS:                    return ("보너스");
      case DEAL_TYPE_COMMISSION:               return ("추가 수수료");
      case DEAL_TYPE_COMMISSION_DAILY:         return ("일별 수수료");
      case DEAL_TYPE_COMMISSION_MONTHLY:       return ("월별 수수료");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY:   return ("일별 에이전트 수수료");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return ("월별 에이전트 수수료");
      case DEAL_TYPE_INTEREST:                 return ("이자율");
      case DEAL_TYPE_BUY_CANCELED:             descr="취소된 매입 거래"; break;
      case DEAL_TYPE_SELL_CANCELED:            descr="취소된 매수 거래"; break;
     }
   descr=StringFormat("%s %G %s (order #%d, position ID %d)",
                      descr,  // 현재 설명
                      volume, // 거래량
                      symbol, // 거래 심볼
                      ticket, // 거래를 야기한 주문의 티켓
                      pos_ID  // 거래가 포함된 위치의 ID
                      );
   return(descr);
//---
  }
```

참고 항목

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), Deal Properties

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetDouble

거래의 요청된 속성을 반환합니다. 거래 속성은 double 타입이어야 합니다. 이 함수의 변수는 두 가지가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,     // 티켓
   ENUM_DEAL_PROPERTY_DOUBLE  property_id        // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 True 또는 False를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 레퍼런스로 전달되는 대상 변수에 배치됩니다.

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,     // 티켓
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,       // 속성 식별자
   double&                    double_var         // 여기서 속성 값을 수락합니다
   );
```

매개 변수

ticket_number

[in] 거래 티켓.

property_id

[in] 딜 속성 식별자. 이 값은 ENUM_DEAL_PROPERTY_DOUBLE 열거값 중 하나일 수 있습니다.

double_var

[out] 요청된 속성 값을 수락하는 double 유형의 변수.

반환값

double 타입의 값.

참고

주문, 딜 및 포지션을 혼동하지 마십시오. 각 거래는 주문 실행의 결과이고 각 포지션은 하나 이상의 거래의 요약 결과입니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래와 주문 히스토리 요청
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- 계정 히스토리의 거래 목록에 의한 루프
   int total=HistoryDealsTotal();
   for(int i=0; i<total; i++)
     {
      //--- 다음 거래 티켓 가져오기(해당 속성을 가져오기 위해 거래가 자동으로 선택됨)
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;

      //--- 거래의 유형과 방향을 가져오고 선택한 거래의 실제 속성 목록에 대한 헤더를 표시합니다.
      string type=DealTypeDescription((ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DE
      string entry=DealEntryDescription((ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket,
      PrintFormat("Double properties of an deal %s entry %s #%I64u:", type, entry, tic

      //--- 헤더 아래에 선택한 거래의 실제 속성을 모두 인쇄합니다.
      HistoryDealPropertiesDoublePrint(ticket, 12);
     }
   /*
   real:
   Double properties of an deal Buy entry In #2785070622:
   Volume:     0.50
   Price:      1.10480
   Commission: 0.00
   Swap:       0.00
   Profit:     0.00 USD
   Fee:        0.00
   StopLoss:   0.00000
   TakeProfit: 0.00000
   Double properties of an deal Sell entry Out #2785071538:
   Volume:     0.50
   Price:      1.10491
   Commission: 0.00
   Swap:       0.00
   Profit:     5.50 USD
   Fee:        0.00
   StopLoss:   0.00000
   TakeProfit: 0.00000
   */
  }
//+------------------------------------------------------------------+
//| Display real properties of the selected deal in the journal      |
//+------------------------------------------------------------------+
void HistoryDealPropertiesDoublePrint(const ulong ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   double value=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- get the deal symbol, profit currency and the number of decimal places for the sy
   string symbol   = HistoryDealGetString(ticket, DEAL_SYMBOL);
   string currency= SymbolInfoString(symbol, SYMBOL_CURRENCY_PROFIT);
   int    digits   = (int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);

//--- 헤더 텍스트와 헤더 필드의 너비를 정의합니다.
//--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 10이 됩니다.
   header="Volume:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- get and display the deal volume with the specified header width in the journal
   if(!HistoryDealGetDouble(ticket, DEAL_VOLUME, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- display the deal price in the journal
   header="Price:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_PRICE, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- display the deal commission in the journal
   header="Commission:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_COMMISSION, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- display the accumulated swap in the journal when closing
   header="Swap:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_SWAP, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);

//--- display the deal financial result in the journal
   header="Profit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_PROFIT, value))
      return;
   PrintFormat("%-*s%-.2f %s", w, header, value, currency);

//--- display the deal fee in the journal
   header="Fee:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_FEE, value))
      return;
   PrintFormat("%-*s%-.2f", w, header, value);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 저널에 StopLoss 레벨을 표시
   header="StopLoss:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_SL, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);

//--- 저널에 TakeProfit 레벨을 표시
   header="TakeProfit:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetDouble(ticket, DEAL_TP, value))
      return;
   PrintFormat("%-*s%-.*f", w, header, digits, value);
  }
//+------------------------------------------------------------------+
//| 거래 타입 설명을 반환                                            |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                    :  return("Buy");
      case DEAL_TYPE_SELL                   :  return("Sell");
      case DEAL_TYPE_BALANCE                :  return("Balance");
      case DEAL_TYPE_CREDIT                 :  return("Credit");
      case DEAL_TYPE_CHARGE                 :  return("Additional charge");
      case DEAL_TYPE_CORRECTION             :  return("Correction");
      case DEAL_TYPE_BONUS                  :  return("Bonus");
      case DEAL_TYPE_COMMISSION             :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY       :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY     :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY:  return("Monthly agent commission");
      case DEAL_TYPE_INTEREST               :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED           :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED          :  return("Canceled sell deal");
      case DEAL_DIVIDEND                    :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED            :  return("Franked (non-taxable) dividend
      case DEAL_TAX                         :  return("Tax charges");
      default                               :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| 포지션 변경 메서드 반환                                          |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      case DEAL_ENTRY_IN       :  return("In");
      case DEAL_ENTRY_OUT      :  return("Out");
      case DEAL_ENTRY_INOUT    :  return("Reverce");
      case DEAL_ENTRY_OUT_BY   :  return("Out by");
      case DEAL_ENTRY_STATE    :  return("Status record");
      default                  :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

참고 항목

HistorySelect(), HistoryDealsTotal(), HistoryDealGetTicket(), Deal Properties

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetInteger

거래의 요청된 속성을 반환합니다. 거래 속성은 datetime, int 유형이어야 합니다. 이 함수의 변수는 두 가지가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
long  HistoryDealGetInteger(
   ulong                      ticket_number,     // 티켓
   ENUM_DEAL_PROPERTY_INTEGER  property_id       // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 True 또는 False를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 레퍼런스로 전달되는 대상 변수에 배치됩니다.

```
bool  HistoryDealGetInteger(
   ulong                      ticket_number,     // 티켓
   ENUM_DEAL_PROPERTY_INTEGER  property_id,      // 속성 식별자
   long&                      long_var           // 여기서 속성 값을 수락합니다
   );
```

Parameter

ticket_number

[in] 거래 티켓.

property_id

[in] 딜 속성 식별자. 이 값은 ENUM_DEAL_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

long_var

[out] 요청된 속성 값을 수락하는 long 타입의 변수.

반환값

long 타입 값.

참고

주문, 딜 및 포지션을 혼동하지 마십시오. 각 거래는 주문 실행의 결과이고 각 포지션은 하나 이상의 거래의 요약 결과입니다.

예:

```
//+------------------------------------------------------------------+
//| 거래 함수                                                        |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- 지난 거래 내역에서 최근 티켓 수령
   ulong last_deal=GetLastDealTicket();
   if(HistoryDealSelect(last_deal))
     {
거래 실행 시간(밀리초)(1970.01.01 이후)       //--- 1970년 1월 1일 이후 밀리세컨드 단위의 거래
```

© 2000-2025, MetaQuotes Ltd.

```
      long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
      PrintFormat("Deal #%d DEAL_TIME_MSC=%i64 => %s",
                  last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
     }
   else
      PrintFormat("HistoryDealSelect() failed for #%d. Eror code=%d",
                  last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------+
//| 내역의 마지막 딜 티켓 또는 -1을 반환합니다                          |
//+------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
//--- 지난 7일 동안의 내역을 요청합니다
   if(!GetTradeHistory(7))
     {
      //--- 요청 실패에 대해 알림 후 -1 반환
      Print(__FUNCTION__," HistorySelect() returned false");
      return -1;
     }
//---
   ulong first_deal,last_deal,deals=HistoryDealsTotal();
//--- 주문이 있다면 작업 수행
   if(deals>0)
     {
      Print("Deals = ",deals);
      first_deal=HistoryDealGetTicket(0);
      PrintFormat("first_deal = %d",first_deal);
      if(deals>1)
        {
         last_deal=HistoryDealGetTicket((int)deals-1);
         PrintFormat("last_deal = %d",last_deal);
         return last_deal;
        }
      return first_deal;
     }
//--- 거래를 찾을 수 없음, -1 반환
   return -1;
  }
//+------------------------------------------------------------------------+
//| 최근 일들 동안의 내역을 요청하고 장애가 발생한 경우 False를 반환합니다  |
//+------------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- 거래 내역을 의뢰할 주간의 기간을 설정
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 요청을 하고 결과 확인
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
      return false;
     }
//--- 이력 수령 성공
   return true;
  }
```

참고 항목

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Deal Properties

© 2000-2025, MetaQuotes Ltd.

HistoryDealGetString

거래의 요청된 속성을 반환합니다. 거래 속성은 string 유형이어야 합니다. 이 함수의 변수는 두 가지가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
string  HistoryDealGetString(
   ulong                      ticket_number,     // 티켓
   ENUM_DEAL_PROPERTY_STRING  property_id        // 속성 식별자
   );
```

2. 함수의 성공 여부에 따라 True 또는 False를 반환합니다. 성공하면 속성 값이 마지막 매개 변수에서 레퍼런스로 전달되는 대상 변수에 배치됩니다.

```
bool  HistoryDealGetString(
   ulong                      ticket_number,     // 티켓
   ENUM_DEAL_PROPERTY_STRING  property_id,       // 속성 식별자
   string&                    string_var         // 여기서 속성 값을 수락합니다
   );
```

매개 변수

ticket_number

[in] 거래 티켓.

property_id

[in] 딜 속성 식별자. 이 값은 ENUM_DEAL_PROPERTY_STRING 열거값 중 하나일 수 있습니다.

string_var

[out] 요청된 속성의 값을 수락하는 string 유형의 변수.

반환값

string 타입의 변수.

참고

주문, 딜 및 포지션을 혼동하지 마십시오. 각 거래는 주문 실행의 결과이고 각 포지션은 하나 이상의 거래의 요약 결과입니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 거래와 주문 히스토리 요청
   if(!HistorySelect(0, TimeCurrent()))
     {
      Print("HistorySelect() failed. Error ", GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }

//--- 계정 히스토리의 거래 목록에 의한 루프
   int total=HistoryDealsTotal();
   for(int i=0; i<total; i++)
     {
      //--- 다음 거래 티켓 가져오기(해당 속성을 가져오기 위해 거래가 자동으로 선택됨)
      ulong ticket=HistoryDealGetTicket(i);
      if(ticket==0)
         continue;

      //--- 거래의 유형과 방향을 가져오고 선택한 거래의 실제 속성 목록에 대한 헤더를 표시합니다.
      string type=DealTypeDescription((ENUM_DEAL_TYPE)HistoryDealGetInteger(ticket, DE
      string entry=DealEntryDescription((ENUM_DEAL_ENTRY)HistoryDealGetInteger(ticket,
      PrintFormat("String properties of an deal %s entry %s #%I64u:", type, entry, tic

      //--- 헤더 아래에 선택한 거래의 실제 속성을 모두 인쇄합니다.
      HistoryDealPropertiesStringPrint(ticket, 13);
     }
   /*
   결과:
   String properties of an deal Buy entry In #2785021084:
   Symbol:       EURUSD
   Comment:      Test PositionGetString
   Extarnal ID:
   String properties of an deal Buy entry Out #2497993663:
   Symbol:       EURUSD
   Comment:      [tp 1.08639]
   Extarnal ID:
   */
  }
//+------------------------------------------------------------------+
//| 저널에서 선택한 거래의 문자열 속성을 표시합니다.                          |
//+------------------------------------------------------------------+
void HistoryDealPropertiesStringPrint(const ulong ticket, const uint header_width=0)
  {
   uint   w=0;
   string header="";
   string value ="";

//--- 헤더 텍스트와 헤더 필드의 너비를 정의합니다.
//--- 헤더 너비가 0과 같은 함수에 전달되면 너비는 헤더 줄 크기 + 10이 됩니다.
   header="Symbol:";
   w=(header_width==0 ? header.Length()+1 : header_width);
//--- 저널에 지정된 헤더 너비를 가진 거래 심볼을 가져오고 표시합니다.
   if(!HistoryDealGetString(ticket, DEAL_SYMBOL, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 거래 코멘트를 저널에 표시합니다.
   header="Comment:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetString(ticket, DEAL_COMMENT, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);

//--- 외부 거래 시스템에 거래 ID를 표시합니다.
   header="Extarnal ID:";
   w=(header_width==0 ? header.Length()+1 : header_width);
   if(!HistoryDealGetString(ticket, DEAL_EXTERNAL_ID, value))
      return;
   PrintFormat("%-*s%-s", w, header, value);
  }
//+------------------------------------------------------------------+
//| 거래 타입 설명을 반환                                            |
//+------------------------------------------------------------------+
string DealTypeDescription(const ENUM_DEAL_TYPE type)
  {
   switch(type)
     {
      case DEAL_TYPE_BUY                     :  return("Buy");
      case DEAL_TYPE_SELL                    :  return("Sell");
      case DEAL_TYPE_BALANCE                 :  return("Balance");
      case DEAL_TYPE_CREDIT                  :  return("Credit");
      case DEAL_TYPE_CHARGE                  :  return("Additional charge");
      case DEAL_TYPE_CORRECTION              :  return("Correction");
      case DEAL_TYPE_BONUS                   :  return("Bonus");
      case DEAL_TYPE_COMMISSION              :  return("Additional commission");
      case DEAL_TYPE_COMMISSION_DAILY        :  return("Daily commission");
      case DEAL_TYPE_COMMISSION_MONTHLY      :  return("Monthly commission");
      case DEAL_TYPE_COMMISSION_AGENT_DAILY  :  return("Daily agent commission");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY:  return("Monthly agent commission");
      case DEAL_TYPE_INTEREST                :  return("Interest rate");
      case DEAL_TYPE_BUY_CANCELED            :  return("Canceled buy deal");
      case DEAL_TYPE_SELL_CANCELED           :  return("Canceled sell deal");
      case DEAL_DIVIDEND                     :  return("Dividend operations");
      case DEAL_DIVIDEND_FRANKED             :  return("Franked (non-taxable) dividend
      case DEAL_TAX                          :  return("Tax charges");
      default                                :  return("Unknown deal type: "+(string)t
     }
  }
//+------------------------------------------------------------------+
//| 포지션 변경 메서드 반환                                          |
//+------------------------------------------------------------------+
string DealEntryDescription(const ENUM_DEAL_ENTRY entry)
  {
   switch(entry)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      case DEAL_ENTRY_IN      :  return("In");
      case DEAL_ENTRY_OUT     :  return("Out");
      case DEAL_ENTRY_INOUT   :  return("Reverce");
      case DEAL_ENTRY_OUT_BY  :  return("Out by");
      case DEAL_ENTRY_STATE   :  return("Status record");
      default                 :  return("Unknown deal entry: "+(string)entry);
     }
  }
```

참고 항목

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), Deal Properties

© 2000-2025, MetaQuotes Ltd.

売買シグナル

이것은 거래 시그널을 관리하기 위한 함수 그룹입니다. 이 함수들은 다음을 허용합니다:

- 거래 시그널에 대한 정보를 얻고, 복제를 위해 이용가능하며,
- 신호 복제 설정을 가져오고 설정하며,
- MQL5 언어 함수를 사용하여 시그널 복제를 구독하거나 구독을 취소합니다.

함수	기능
SignalBaseGetDouble	선택한 시그널에 대한 double 유형 속성 값을 반환합니다.
SignalBaseGetInteger	선택한 시그널에 대한 integer 타입 속성 값을 반환합니다.
SignalBaseGetString	선택한 시그널에 대한 string 타입 속성 값을 반환합니다.
SignalBaseSelect	시그널들 중에서 한 시그널을 선택합니다. 시그널을 사용하여 추가 작업을 위해 터미널에서 사용할 수 있습니다.
SignalBaseTotal	터미널에서 사용할 수 있는 총 시그널 양을 반환합니다.
SignalInfoGetDouble	시그널 복제 설정의 double 타입 속성 값을 반환합니다.
SignalInfoGetInteger	시그널 복제 설정의 integer 타입 속성 값을 반환합니다.
SignalInfoGetString	시그널 복제 설정의 string 타입 속성 값을 반환합니다.
SignalInfoSetDouble	시그널 복제 설정의 double 타입 속성 값을 설정합니다.
SignalInfoSetInteger	시그널 복제 설정의 integer 타입 속성 값을 설정합니다.
SignalSubscribe	트레이딩 시그널에 구독합니다
SignalUnsubscribe	구독 취소

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetDouble

선택한 시그널에 대한 double 타입의 속성 값을 반환합니다.

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id,     // 속성 식별자
   );
```

Parameter

property_id

[in] 시그널 속성 식별자. 값은 ENUM_SIGNAL_BASE_DOUBLE 열거값 중 하나일 수 있습니다.

반환값

선택한 시그널의 double 타입의 속성 값.

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetInteger

선택한 시그널에 대한 integer 타입의 속성 값을 반환합니다.

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id,     // 속성 식별자
   );
```

Parameter

property_id

[in] 시그널 속성 식별자. 이 값은 ENUM_SIGNAL_BASE_INTEGER 열거값 중 하나일 수 있습니다.

반환값

선택한 시그널의 integer 타입 속성 값.

© 2000-2025, MetaQuotes Ltd.

SignalBaseGetString

선택한 시그널에 대해 string 타입의 속성 값을 반환합니다.

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id,     // 속성 식별자
   );
```

Parameter

property_id

[in] 시그널 속성 식별자. 이 값은 ENUM_SIGNAL_BASE_STRING 열거값 중 하나일 수 있습니다.

반환값

선택한 시그널의 string 타입의 속성 값.

© 2000-2025, MetaQuotes Ltd.

SignalBaseSelect

시그널들 중에서 한 시그널을 선택합니다. 시그널로 추가 작업을 위해 터미널에서 사용할 수 있습니다.

```
bool  SignalBaseSelect(
   int      index     // 시그널 인덱스
   );
```

Parameter

index

[in] 트레이딩 시그널 기준 시그널 인덱스.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

예:

```
void OnStart()
  {
//--- 터미널의 총 시그널 양을 수신
   int total=SignalBaseTotal();
//--- 모든 시그널 처리
   for(int i=0;i<total;i++)
     {
      //--- 인덱스로 시그널 선택
      if(SignalBaseSelect(i))
        {
         //--- 시그널 속성 취득
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);           // 시그널 id
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);         // 핍(pip) 수익
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // 구독자 수
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);          // 시그널명
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);         // 시그널 가격
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);      // 시그널 통화
         //--- 모든 수익성 있는 무료 시그널을 구독자에게 출력
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%d",id,
        }
      else PrintFormat("SignalBaseSelect 요청 중 에러. Error code=%d",GetLastError());
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SignalBaseTotal

터미널에서 사용할 수 있는 총 시그널 양을 반환합니다.

```
int  SignalBaseTotal();
```

반환값

터미널에서 사용할 수 있는 총 시그널 양.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetDouble

시그널 복사 설정의 double형 속성 값을 반환합니다.

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id,     // 속성 식별자
   );
```

Parameter

property_id

[in] 시그널 복제 설정 속성 식별자. 이 값은 ENUM_SIGNAL_INFO_DOUBLE 열거값 중 하나일 수 있습니다.

반환값

시그널 복제 설정의 double 타입의 속성 값.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetInteger

시그널 복제 설정의 integer 타입 속성 값을 반환합니다.

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // 속성 식별자
   );
```

Parameter

property_id

[in] 시그널 복제 설정 속성 식별자. 그 값은 ENUM_SIGNAL_INFO_INTEGER 열거값 중 하나일 수 있습니다.

반환값

시그널 복제 설정의 integer 타입의 속성 값.

© 2000-2025, MetaQuotes Ltd.

SignalInfoGetString

시그널 복제 설정의 string 타입의 속성 값을 반환합니다.

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id,     // 속성 식별자
   );
```

Parameter

property_id

[in] 시그널 복제 설정 속성 식별자. 그 값은 ENUM_SIGNAL_INFO_STRING 열거값 중 하나일 수 있습니다.

반환값

시그널 복제 설정의 string 타입의 속성 값.

© 2000-2025, MetaQuotes Ltd.

SignalInfoSetDouble

시그널 복제 설정의 double 타입의 속성 값을 설정합니다.

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE      property_id,     // 속성 식별자
   double                       value            // 새 값
   );
```

Parameter

property_id

[in] 시그널 복제 설정 속성 식별자. 이 값은 ENUM_SIGNAL_INFO_DOUBLE 열거값 중 하나일 수 있습니다.

value

[in] 시그널 복제 설정 속성 값.

반환값

속성이 변경된 경우 true를 반환하고 그렇지 않으면 false를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

© 2000-2025, MetaQuotes Ltd.

SignalInfoSetInteger

시그널 복제 설정의 integer 타입의 속성 값을 설정합니다.

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // 속성 식별자
   long                         value            // 새 값
   );
```

Parameter

property_id

[in] 시그널 복제 설정 속성 식별자. 그 값은 ENUM_SIGNAL_INFO_INTEGER 열거값 중 하나일 수 있습니다.

value

[in] 시그널 복제 설정 속성 값.

반환값

속성이 변경된 경우 true를 반환하고 그렇지 않으면 false를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

© 2000-2025, MetaQuotes Ltd.

SignalSubscribe

거래 시그널을 구독합니다.

```
bool  SignalSubscribe(
   long     signal_id     // 시그널 id
   );
```

Parameter

signal_id

[in] 시그널 식별자.

반환값

구독이 성공적이면 true를 반환하고 그렇지 않으면 false를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

© 2000-2025, MetaQuotes Ltd.

SignalUnsubscribe

구독 취소.

```
bool  SignalUnsubscribe();
```

반환값

구독이 성공적으로 취소되면 true를 반환하고 그렇지 않으면 false를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

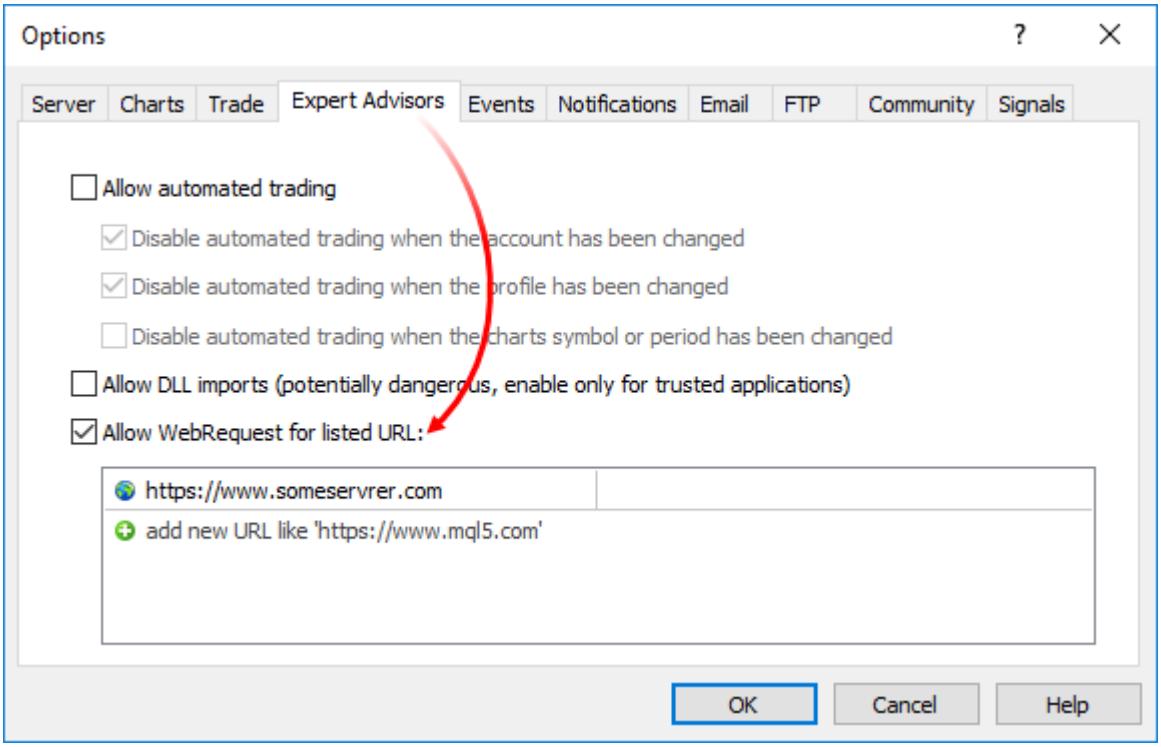
© 2000-2025, MetaQuotes Ltd.

네트워크 함수

MQL5 프로그램은 원격 서버와 데이터를 교환하고 FTP를 통해 푸시 알림, 이메일 및 데이터를 전송할 수 있습니다.

- The 소켓* 함수 그룹을 사용하면 시스템 소켓을 통해 원격 호스트와의 TCP 연결(보안 TLS 포함)을 설정할 수 있습니다. 작동 원칙은 간단합니다: 소켓을 생성하고, 서버에 연결한 후 데이터 읽기 및 쓰기를 시작하십시오.
- WebRequest 기능은 웹 자원과 함께 작동하도록 설계되었으며, 쉽게 HTTP 요청(GET 및 POST 포함)을 전송할 수 있습니다.
- SendFTP, SendMail 및 SendNotification은 파일, 이메일 및 모바일 알림을 전송하기 위한 보다 간단한 함수들입니다.

최종 사용자 보안을 위해 허용된 IP 주소 목록이 클라이언트 터미널 측에 구현됩니다. 목록에는 소켓* 및 WebRequest 기능을 통해 MQL5 프로그램이 연결할 수 있는 IP 주소가 포함되어 있습니다. 예를 들어, 프로그램이 https://www.someserver.com에 연결해야 하는 경우, 이 주소는 목록에 있는 터미널 사용자가 명시적으로 표시해야 합니다. 주소를 프로그래밍 방식으로 추가할 수 없습니다.



MQL5 프로그램에 명시적 메시지를 추가하여 추가 구성의 필요성을 사용자에게 알립니다. #property description, Alert or Print를 통해 이 작업을 수행할 수 있습니다.

함수	기능
SocketCreate	지정된 플래그가 있는 소켓을 만들고 핸들을 반환합니다
SocketClose	소켓 닫기
SocketConnect	시간 초과 제어를 사용하여 서버에 연결합니다
SocketIsConnected	소켓이 현재 연결되어 있는지 확인합니다
SocketIsReadable	소켓에서 읽어 올 수 있는 바이트의 번호를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

함수	기능
SocketIsWritable	현재 소켓에 데이터를 쓸 수 있는지 확인합니다
SocketTimeouts	소켓 시스템 개체에 대한 데이터 수신 및 전송 시간에 대한 초과 설정을 합니다
SocketRead	소켓에서 데이터를 읽습니다
SocketSend	소켓에 데이터를 작성합니다
SocketTlsHandshake	TLS Handshake 프로토콜을 통해 지정된 호스트에 대한 보안 TLS(SSL) 연결을 시작합니다
SocketTlsCertificate	네트워크 연결을 보호하는 데 사용되는 인증서의 데이터를 가져옵니다
SocketTlsRead	보안 TLS 연결에서 데이터를 읽습니다
SocketTlsReadAvailable	보안 TLS 연결에서 사용 가능한 모든 데이터를 읽습니다
SocketTlsSend	보안 TLS 연결을 통해 데이터를 전송합니다
WebRequest	지정된 서버로 HTTP 요청을 전송합니다
SendFTP	FTP 탭에 지정된 주소로 파일을 전송합니다
SendMail	옵션 창의 이메일 탭에 지정된 주소로 이메일을 발송합니다
SendNotification	MetaQuotes ID가 알림 탭에 지정된 모바일 단말기에 푸시 알림을 보냅니다

© 2000-2025, MetaQuotes Ltd.

SocketCreate

지정된 플래그가 있는 소켓을 만들고 핸들을 반환합니다.

```
int  SocketCreate(
   uint     flags         // 플래그
   );
```

Parameter

flags

[in] 소켓 작업 모드를 정의하는 플래그 조합입니다. 현재 하나의 플래그만 지원됩니다. — SOCKET_DEFAULT.

반환값

소켓 생성에 성공한 경우, handle을 반환하며, 그렇지않으면 INVALID_HANDLE을 반환합니다.

참고

사용하지 않는 소켓에서 컴퓨터 메모리를 확보하려면, SocketClose 을 호출하십시오.

하나의 MQL5 프로그램에서 최대 128개의 소켓을 생성할 수 있습니다. 제한 초과시, 5271 에러 (ERR_NETSOCKET_TOO_MANY_OPENED)가 _LastError에 기록됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "터미널 설정에서 허용된 주소 목록에 주소를 추가하여 예제가 작동하도록 합
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| 서버로 명령 전송                                                 |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- 포트 443을 통해 보안 TLS 연결을 사용하는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 표준 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 응답을 읽습니다                                             |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- 소켓이 여전히 존재하지만 제한 시간보다 길지 않을 때까지 소켓에서 데이터를 읽습니다
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 연결의 보안 여부에 따라 다양한 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 반응을 분석합니다
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 출력합니다
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP 응답 헤더 수신:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   int socket=SocketCreate();
//--- 핸들 체크
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든 것이 문제 없다면 연결합니다
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("다음에 대한 연결 설정 ",주소,":",포트);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 인증서로 연결이 보안된 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS 인증서:");
            Print("    소유자:  ",subject);
            Print("    발행인:  ",issuer);
            Print("    숫자:      ",serial);
            Print("    Print: ",thumbprint;
            Print("    만료: ",expiration;
            ExtTLS=true;
           }
         //--- GET 요청을 서버로 전송합니다
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET 요청 발송됨");
            //--- 반응을 읽습니다
            if(!HTTPRecv(socket,1000))
               Print("응답 가져오기 실패, error ",GetLastError());
           }
         else
            Print("응답 가져오기 실패, error ",GetLastError());
        }
      else
        {
         Print("다음에 연결 ",주소,":",포트," 실패, error ",GetLastError());
        }
      //--- 이용 후 소켓을 닫습니다
      SocketClose(socket);
     }
   else
      Print("소켓 생성 실패, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketClose

소켓 닫기.

```
bool  SocketClose(
   const int  socket      // 소켓 처리
   );
```

Parameter

socket

[in] 소켓 처리가 닫힙니다. SocketCreate 함수로 소켓 처리 기능이 복귀됩니다. 잘못된 핸들이 전달된 경우, 5270 에러(ERR_NETSOCKET_INVALIDHANDLE)가 _LastError에 작성됩니다.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

SocketConnect를 통한 연결이 이전에 소켓에 대해 생성된 경우, 소켓은 중단됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "터미널 설정에서 허용된 주소 목록에 주소를 추가하여 예제가 작동하도록 합
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| 서버로 명령 전송                                                 |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 포트 443을 통해 보안 TLS 연결을 사용하는 경우
```

© 2000-2025, MetaQuotes Ltd.

```
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 표준 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 응답을 읽습니다                                             |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- 소켓이 여전히 존재하지만 제한 시간보다 길지 않을 때까지 소켓에서 데이터를 읽습니다
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 연결의 보안 여부에 따라 다양한 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 반응을 분석합니다
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 출력합니다
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP 응답 헤더 수신:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 핸들 체크
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든 것이 문제 없다면 연결합니다
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("다음에 대한 연결 설정 ",주소,":",포트);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 인증서로 연결이 보안된 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS 인증서:");
            Print("    소유자:  ",subject);
            Print("    발행인:  ",issuer);
            Print("    숫자:      ",serial);
            Print("    Print: ",thumbprint;
            Print("    만료: ",expiration;
            ExtTLS=true;
           }
         //--- GET 요청을 서버로 전송합니다
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET 요청 발송됨");
            //--- 반응을 읽습니다
            if(!HTTPRecv(socket,1000))
               Print("응답 가져오기 실패, error ",GetLastError());
           }
         else
            Print("응답 가져오기 실패, error ",GetLastError());
        }
      else
        {
         Print("다음에 연결 ",주소,":",포트," 실패, error ",GetLastError());
        }
      //--- 이용 후 소켓을 닫습니다
      SocketClose(socket);
     }
   else
      Print("소켓 생성 실패, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketConnect

타임아웃 컨트롤을 사용하여 서버에 연결합니다.

```
bool  SocketConnect(
   int           socket,              // 소켓
   const string  server,              // 연결 주소
   uint          port,                // 연결 포트
   uint          timeout_receive_ms   // 연결 타임아웃
   );
```

Parameter

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 전달된 경우, 5270 에러 (ERR_NETSOCKET_INVALIDHANDLE)가 _LastError에 작성됩니다.

server

[in] 연결할 서버의 도메인 이름 또는 IP 주소.

port

[in] 연결 포트 번호.

timeout_receive_ms

[in] 밀리세컨드 단위의 타임아웃 연결. 해당 시간 간격 내에 연결되지 않으면 시도가 중지됩니다.

반환값

연결에 성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

클라이언트 터미널(Tools→Options→Expert Advisors) 측에서 허용된 연결 주소 목록에 연결 주소를 추가해야 합니다.

연결 실패시, 5272 오류(ERR_NETSOCKET_CANNOT_CONNECT)가 _LastError에 작성됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "터미널 설정에서 허용된 주소 목록에 주소를 추가하여 예제가 작동하도록 합
#property script_show_inputs
```

© 2000-2025, MetaQuotes Ltd.

```
input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| 서버로 명령 전송                                                 |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 포트 443을 통해 보안 TLS 연결을 사용하는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 표준 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 응답을 읽습니다                                             |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- 소켓이 여전히 존재하지만 제한 시간보다 길지 않을 때까지 소켓에서 데이터를 읽습니다
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 연결의 보안 여부에 따라 다양한 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 반응을 분석합니다
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 출력합니다
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP 응답 헤더 수신:");
               Print(StringSubstr(result,0,header_end));
```

© 2000-2025, MetaQuotes Ltd.

```
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- 핸들 체크
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든 것이 문제 없다면 연결합니다
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("다음에 대한 연결 설정 ",주소,":",포트);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 인증서로 연결이 보안된 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS 인증서:");
            Print("    소유자:  ",subject);
            Print("    발행인:  ",issuer);
            Print("    숫자:      ",serial);
            Print("    Print: ",thumbprint;
            Print("    만료: ",expiration;
            ExtTLS=true;
           }
         //--- GET 요청을 서버로 전송합니다
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET 요청 발송됨");
            //--- 반응을 읽습니다
            if(!HTTPRecv(socket,1000))
               Print("응답 가져오기 실패, error ",GetLastError());
           }
         else
            Print("응답 가져오기 실패, error ",GetLastError());
        }
      else
        {
         Print("다음에 연결 ",주소,":",포트," 실패, error ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
         }
       //--- 이용 후 소켓을 닫습니다
       SocketClose(socket);
      }
    else
       Print("소켓 생성 실패, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketIsConnected

소켓이 현재 연결되어 있는지 확인합니다.

```
bool  SocketIsConnected(
   const int  socket      // 소켓 처리
   );
```

매개 변수

socket

[in] 소켓 핸들이 SocketCreate() 함수에 의해 반환됩니다. 잘못된 핸들이 _LastError로 전달될 시, 5270 에러(ERR_NETSOCKET_INVALIDHANDLE)가 활성화됩니다.

반환값

소켓이 연결된 경우 true를 반환하고, 그렇지 않은 경우 false를 반환합니다.

참고

SocketIsConnected() 함수를 사용하면 현재 소켓 연결 상태를 확인할 수 있습니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                            SocketIsConnected.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "예제가 작동할 수 있도록 터미널 설정에서 허용된 주소 목록에 추가하세요"
#property script_show_inputs

input string Address    ="www.mql5.com";
input int    Port       =80;
input bool   CloseSocket=true;
bool         ExtTLS     =false;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 소켓을 생성하고 핸들을 수신
   int socket=SocketCreate();
//--- 핸들을 확인하세요
   if(socket!=INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      //--- 모든게 준비되면 연결하세요
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 연결이 인증서로 보호되는 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:      ",subject);
            Print("   Issuer:     ",issuer);
            Print("   Number:     ",serial);
            Print("   Print:      ",thumbprint);
            Print("   Expiration: ",expiration);
            ExtTLS=true;
           }
         //--- GET 요청을 서버에 보내세요
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET request sent");
            //--- 회신 읽기
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         PrintFormat("Connection to %s:%d failed, error %d",Address,Port,GetLastError
        }
      //--- 플래그가 설정되어 있으면 사용 후 소켓을 닫습니다.
      if(CloseSocket)
         SocketClose(socket);
     }
   else
      Print("Failed to create a socket, error ",GetLastError());

//--- 서버 연결을 확인하세요
   bool connected=SocketIsConnected(socket);
//--- 연결이 되어 있지 않으면 동작을 종료합니다
   if(!connected)
     {
      Print("No connection to server");
     }
//--- 만약 서버 연결이 되어 있다면
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      Print("Connection to the server is available\nThe connection needs to be closed.
      //--- 소켓을 닫고 연결 상태를 다시 확인하세요
      SocketClose(socket);
      connected=SocketIsConnected(socket);
     }

//--- 현재 서버 연결 상태를 표시합니다
   Print("Currently connected: ",(connected ? "opened" : "closed"));
   /*
   result in case CloseSocket = true:
   No connection to server
   Currently connected: closed

   result in case CloseSocket = false:
   Connection to the server is available
   The connection needs to be closed. Closing
   Currently connected: closed
   */
  }
//+------------------------------------------------------------------+
//| 서버에 명열을 보냅니다                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
//--- 문자열을 문자 배열로 변환하고 종료 0을 버립니다.
   char req[];
   int  len=StringToCharArray(request,req)-1;

   if(len<0)
      return(false);
//--- 포트 443을 통한 보안 TLS 연결이 사용되는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 일반 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 회신 읽기                                                   |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;
//--- 데이터가 있는 동안 소켓에서 데이터를 읽습니다. 그러나 시간 초과보다 길지 않도록 합니다.
   do
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      uint len=SocketIsReadable(socket);

      if(len)
        {
         int rsp_len;
         //--- 연결이 안전한지 여부에 따라 다른 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout_ms);
         //--- 응답 구문 분석
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 표시
            int header_end=StringFind(result,"\r\n\r\n");

            if(header_end>0)
              {
               Print("HTTP answer header received:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
            //--- 읽기 제한 시간 만료 시간 업데이트
            timeout_check=GetTickCount64()+timeout_ms;
           }
        }
     }
   while(GetTickCount64()<timeout_check && !IsStopped());

   return(false);
  }
```

참고 항목

SocketConnect, SocketIsWritable, SocketCreate, SocketClose

© 2000-2025, MetaQuotes Ltd.

SocketIsReadable

소켓에서 읽을 수 있는 바이트 수를 가져옵니다.

```
uint  SocketIsReadable(
   const int  socket      // 소켓 처리
   );
```

Parameter

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 _LastError로 전달될 시, 5270 에러(ERR_NETSOCKET_INVALIDHANDLE)가 활성화됩니다.

반환값

읽을 수 있는 바이트 수입니다. 오류가 발생하면 0이 반환됩니다.

참고

함수 실행시 시스템 소켓에 오류가 발생하면, SocketConnect을 통해 설정된 연결이 중단됩니다.

SocketRead호출 전에, 소켓에 판독할 데이터가 있는지 확인합니다. 그렇지 않으면 데이터가 없는 경우 SocketRead 함수는 프로그램 실행을 지연시키는 timeout_ms 내의 데이터를 기다립니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "터미널 설정에서 허용된 주소 목록에 주소를 추가하여 예제가 작동하도록 합
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| 서버로 명령 전송                                                 |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
//--- 포트 443을 통해 보안 TLS 연결을 사용하는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 표준 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 응답을 읽습니다                                             |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- 소켓이 여전히 존재하지만 제한 시간보다 길지 않을 때까지 소켓에서 데이터를 읽습니다
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 연결의 보안 여부에 따라 다양한 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 반응을 분석합니다
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 출력합니다
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP 응답 헤더 수신:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   int socket=SocketCreate();
//--- 핸들 체크
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든 것이 문제 없다면 연결합니다
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("다음에 대한 연결 설정 ",주소,":",포트);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 인증서로 연결이 보안된 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS 인증서:");
            Print("    소유자:  ",subject);
            Print("    발행인:  ",issuer);
            Print("    숫자:      ",serial);
            Print("    Print: ",thumbprint;
            Print("    만료: ",expiration;
            ExtTLS=true;
           }
         //--- GET 요청을 서버로 전송합니다
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET 요청 발송됨");
            //--- 반응을 읽습니다
            if(!HTTPRecv(socket,1000))
               Print("응답 가져오기 실패, error ",GetLastError());
           }
         else
            Print("응답 가져오기 실패, error ",GetLastError());
        }
      else
        {
         Print("다음에 연결 ",주소,":",포트," 실패, error ",GetLastError());
        }
      //--- 이용 후 소켓을 닫습니다
      SocketClose(socket);
     }
   else
      Print("소켓 생성 실패, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketIsWritable

현재 소켓에 데이터를 쓸 수 있는지 확인합니다.

```
bool  SocketIsWritable(
   const int  socket      // 소켓 처리
   );
```

매개 변수

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 전달된 경우, 5270 에러 (ERR_NETSOCKET_INVALIDHANDLE)가 _LastError에 작성됩니다.

반환값

작성이 가능하면 true를, 아니면 false를 반환합니다.

참고

이 함수를 사용하면 당장 소켓에 데이터를 쓸 수 있는지 여부를 확인할 수 있습니다.

함수 실행시 시스템 소켓에 오류가 발생하면, SocketConnect을 통해 설정된 연결이 중단됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                              SocketIsWritable.mq5 |
//|                                  Copyright 2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "예제가 작동할 수 있도록 터미널 설정에서 허용된 주소 목록에 추가하세요"
#property script_show_inputs

input string Address    ="www.mql5.com";
input int    Port       =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 소켓을 생성하고 핸들을 수신
   int socket=SocketCreate();
//--- 핸들을 확인하세요
   if(socket!=INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
   {
    //--- 모든게 준비되면 연결하세요
    if(SocketConnect(socket,Address,Port,1000))
      {
       PrintFormat("Established connection to %s:%d",Address,Port);

       string   subject,issuer,serial,thumbprint;
       datetime expiration;
       //--- 연결이 인증서로 보호되는 경우 해당 데이터를 표시합니다
       if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
         {
          Print("TLS certificate:");
          Print("   Owner:      ",subject);
          Print("   Issuer:     ",issuer);
          Print("   Number:     ",serial);
          Print("   Print:      ",thumbprint);
          Print("   Expiration: ",expiration);
          ExtTLS=true;
         }
       //--- GET 요청을 서버에 보내세요
       if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
         {
          Print("GET request sent");
          //--- 회신 읽기
          if(!HTTPRecv(socket,1000))
             Print("Failed to get a response, error ",GetLastError());
         }
       else
          Print("Failed to send GET request, error ",GetLastError());
      }
    else
      {
       PrintFormat("Connection to %s:%d failed, error %d",Address,Port,GetLastError
      }
    //--- 현재 시점에 소켓에 데이터를 쓰는 기능을 저널에 표시
    if(SocketIsWritable(socket))
       Print("At the current moment in time, writing data to the socket is possible"
    else
       Print("It is not possible to write data to the socket at the current time");
    //--- 사용 후 소켓 종료
    if(SocketClose(socket))
       Print("Now the socket is closed");
   }
 else
    Print("Failed to create a socket, error ",GetLastError());
 /*
결과:
 At the current moment in time, writing data to the socket is possible
 Socket is closed now
```

© 2000-2025, MetaQuotes Ltd.

```
   */
  }
//+------------------------------------------------------------------+
//| 서버에 명열을 보냅니다                                           |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
//--- 문자열을 문자 배열로 변환하고 종료 0을 버립니다.
   char req[];
   int  len=StringToCharArray(request,req)-1;

   if(len<0)
      return(false);
//--- 포트 443을 통한 보안 TLS 연결이 사용되는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 일반 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 회신 읽기                                                   |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
  {
   char   rsp[];
   string result;
   ulong  timeout_check=GetTickCount64()+timeout_ms;
//--- 데이터가 있는 동안 소켓에서 데이터를 읽습니다. 그러나 시간 초과보다 길지 않도록 합니다.
   do
     {
      uint len=SocketIsReadable(socket);

      if(len)
        {
         int rsp_len;
         //--- 연결이 안전한지 여부에 따라 다른 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout_ms);
         //--- 응답 구문 분석
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 표시
            int header_end=StringFind(result,"\r\n\r\n");

            if(header_end>0)
              {
```

© 2000-2025, MetaQuotes Ltd.

```
                Print("HTTP answer header received:");
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
             //--- 읽기 제한 시간 만료 시간 업데이트
             timeout_check=GetTickCount64()+timeout_ms;
            }
         }
      }
   while(GetTickCount64()<timeout_check && !IsStopped());

   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketTimeouts

소켓 시스템 객체에 대한 데이터 수신 및 전송에 대한 타임아웃을 설정합니다.

```
bool  SocketTimeouts(
   int           socket,              // 소켓
   uint          timeout_send_ms,     // 데이터 전송 타임아웃
   uint          timeout_receive_ms   // 데이터 획득 타임아웃
   );
```

매개 변수

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 전달된 경우, 5270 에러 (ERR_NETSOCKET_INVALIDHANDLE)가 _LastError에 작성됩니다.

timeout_send_ms

[in] 데이터 전송 타임아웃(밀리초).

timeout_receive_ms

[in] 데이터 획득 타임아웃(밀리초)

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

시스템 개체 타임아웃과 SocketRead을 통해 데이터를 읽을 때 설정된 타임아웃을 혼동하지 마십시오. SocketTimeout은 운영 체제의 소켓 개체에 대해 시간 초과를 한 번 설정합니다. 이러한 타임아웃은 이 소켓을 통해 데이터를 읽고 전송하는 모든 함수에 적용됩니다. SocketRead에서는, 타임아웃이 특정한 데이터 읽기 동작을 위해 설정되어 있습니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketTimeouts.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   TIMEOUT_SEND     5000
#define   TIMEOUT_RECEIVE  5000
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart(void)
  {
//--- 소켓을 생성하고 핸들을 수신
   int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- 소켓 시스템 객체와 관련한 데이터 수신 및 전송에 대한 시간 제한을 설정합니다.
   if(SocketTimeouts(socket,TIMEOUT_SEND,TIMEOUT_RECEIVE))
      PrintFormat("timeouts were successfully set",socket,TIMEOUT_SEND,TIMEOUT_RECEIVE
   else
      PrintFormat("SocketTimeouts(%d, %I64d, %I64d) failed. Error %d",socket,TIMEOUT_S
//--- 소켓 종료
   SocketClose(socket);
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketRead

소켓에서 데이터를 읽습니다.

```
int  SocketRead(
   int           socket,              // 소켓
   uchar&        buffer[],            // 소켓에서 데이터를 읽기 위한 버퍼
   uint          buffer_maxlen,       // number of bytes to read
   uint          timeout_ms           // reading timeout
   );
```

Parameter

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 _LastError로 전달될 시, 5270 에러(ERR_NETSOCKET_INVALIDHANDLE)가 활성화됩니다.

buffer

[out] 데이터를 읽을 수 있는 uchar 유형 배열에 대한 레퍼런스입니다. 동적 배열 크기가 읽기 바이트 수만큼 증가합니다. 배열의 크기는 INT_MAX (2147483647)을 초과할 수 없습니다.

buffer_maxlen

[in] *buffer[]* 배열에 읽을 바이트 수. 배열에 맞지 않는 데이터는 소켓에 남아 있습니다. 그들은 다음 SocketRead 호출을 받을 수 있습니다. *buffer_maxlen*은 INT_MAX (2147483647)를 초과할 수 없습니다.

timeout_ms

[in] 데이터 읽기 타임아웃(밀리초). 이 시간 내에 데이터를 얻지 못하면 시도가 중지되고 함수는 -1을 반환합니다.

반환값

성공적이면 읽기 바이트 수를 반환합니다. 오류인 경우, -1이 반환됩니다.

참고

함수 실행시 시스템 소켓에 오류가 발생하면, SocketConnect을 통해 설정된 연결이 중단됩니다.

데이터 읽기 오류가 발생시, 5273 에러(ERR_NETSOCKET_IO_ERROR)가 _LastError에 기록됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "터미널 설정에서 허용된 주소 목록에 주소를 추가하여 예제가 작동하도록 합
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| 서버로 명령 전송                                                 |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 포트 443을 통해 보안 TLS 연결을 사용하는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 표준 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 응답을 읽습니다                                             |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- 소켓이 여전히 존재하지만 제한 시간보다 길지 않을 때까지 소켓에서 데이터를 읽습니다
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 연결의 보안 여부에 따라 다양한 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 반응을 분석합니다
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 출력합니다
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
```

© 2000-2025, MetaQuotes Ltd.

```
                Print("HTTP 응답 헤더 수신:");
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
    while(GetTickCount()<timeout_check && !IsStopped());
    return(false);
   }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- 핸들 체크
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든 것이 문제 없다면 연결합니다
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("다음에 대한 연결 설정 ",주소,":",포트);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 인증서로 연결이 보안된 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS 인증서:");
            Print("    소유자:  ",subject);
            Print("    발행인:  ",issuer);
            Print("    숫자:     ",serial);
            Print("   Print: ",thumbprint;
            Print("    만료: ",expiration;
            ExtTLS=true;
           }
         //--- GET 요청을 서버로 전송합니다
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET 요청 발송됨");
            //--- 반응을 읽습니다
            if(!HTTPRecv(socket,1000))
               Print("응답 가져오기 실패, error ",GetLastError());
           }
         else
            Print("응답 가져오기 실패, error ",GetLastError());
        }
      else
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          Print("다음에 연결 ",주소,":",포트," 실패, error ",GetLastError());
         }
       //--- 이용 후 소켓을 닫습니다
       SocketClose(socket);
      }
    else
       Print("소켓 생성 실패, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

참고 항목

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketSend

소켓에 데이터를 작성하십시오.

```
int  SocketSend(
   int           socket,              // 소켓
   const uchar&  buffer[],            // 데이터 버퍼
   uint          buffer_len           // 버퍼 사이즈
   );
```

Parameter

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 전달된 경우, 5270 에러 (ERR_NETSOCKET_INVALIDHANDLE)가 _LastError에 작성됩니다.

buffer

[in] 소켓으로 전송할 데이터가 있는 uchar 타입 배열을 참조.

buffer_len

[in] '버퍼' 배열 사이즈.

반환값

성공적이면 소켓에 작성된 바이트 수를 반환합니다. 오류인 경우, -1이 반환됩니다.

참고

함수 실행시 시스템 소켓에 오류가 발생하면, SocketConnect을 통해 설정된 연결이 중단됩니다.

데이터 쓰기 오류가 발생시, 5273 에러(ERR_NETSOCKET_IO_ERROR)가 _LastError에 기록됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "터미널 설정에서 허용된 주소 목록에 주소를 추가하여 예제가 작동하도록 합
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 서버로 명령 전송                                                  |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 포트 443을 통해 보안 TLS 연결을 사용하는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 표준 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 응답을 읽습니다                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- 소켓이 여전히 존재하지만 제한 시간보다 길지 않을 때까지 소켓에서 데이터를 읽습니다
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 연결의 보안 여부에 따라 다양한 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 반응을 분석합니다
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 출력합니다
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP 응답 헤더 수신:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- 핸들 체크
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든 것이 문제 없다면 연결합니다
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("다음에 대한 연결 설정 ",주소,":",포트);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 인증서로 연결이 보안된 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS 인증서:");
            Print("    소유자:  ",subject);
            Print("    발행인:  ",issuer);
            Print("    숫자:      ",serial);
            Print("    Print: ",thumbprint;
            Print("    만료: ",expiration;
            ExtTLS=true;
           }
         //--- GET 요청을 서버로 전송합니다
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET 요청 발송됨");
            //--- 반응을 읽습니다
            if(!HTTPRecv(socket,1000))
               Print("응답 가져오기 실패, error ",GetLastError());
           }
         else
            Print("응답 가져오기 실패, error ",GetLastError());
        }
      else
        {
         Print("다음에 연결 ",주소,":",포트," 실패, error ",GetLastError());
        }
      //--- 이용 후 소켓을 닫습니다
      SocketClose(socket);
     }
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("소켓 생성 실패, error ",GetLastError());
   }
//+------------------------------------------------------------------+
```

참고 항목

SocketTimeouts, MathSwap, StringToCharArray

© 2000-2025, MetaQuotes Ltd.

SocketTlsHandshake

TLS Handshake 프로토콜을 통해 지정된 호스트에 대한 보안 TLS(SSL) 연결을 시작합니다. Handshake 중에 클라이언트와 서버는 연결 매개 변수(적용된 프로토콜 버전 및 데이터 암호화 방법)에 동의합니다.

```
bool  SocketTlsHandshake(
   int           socket,              // 소켓
   const string  host                 // 호스트 주소
   );
```

매개 변수

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 전달된 경우, 5270 에러(ERR_NETSOCKET_INVALIDHANDLE)가 _LastError에 작성됩니다.

host

[in] 보안 연결이 설정된 호스트의 주소.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

보안 연결 전, 프로그램은 SocketConnect를 사용하여 호스트와의 표준 TCP 연결을 설정해야 합니다.

보안 연결 실패시, 5274 에러(ERR_NETSOCKET_HANDSHAKE_FAILED)가 _LastError에 기록됩니다.

포트 443에 연결시, 함수를 호출할 필요가 없습니다. SSL(보안 TLS) 연결에 사용되는 표준 TCP 포트입니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                          SocketTlsHandshake.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   SERVER    "smtp.gmail.com"
#define   PORT      587
//+------------------------------------------------------------------+
//| 보안 연결로 수동으로 바꿈                                                  |
//+------------------------------------------------------------------+
bool TlsHandshake(int socket)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 서버 인사말 받기
   string rsp;

   if(!RecvString(socket,rsp))
      return(false);
//--- 서버 접속
   if(!SendString(socket,"EHLO my.domain.com\r\n"))
      return(false);
//--- 지원되는 명령 목록이 포함된 서버 응답을 얻습니다.
   if(!RecvString(socket,rsp))
      return(false);
//--- 인사말 출력
   Print("SERVER: ",rsp);
//--- TLS를 사용하여 안전하지 않은 연결에서 보안 연결로 전환하고 싶다고 서버에 알립니다.
   if(!SendString(socket,"STARTTLS\r\n"))
      return(false);
//--- 서버 회신 수신
   if(!RecvString(socket,rsp))
      return(false);
//--- 예시에서는 TLS로 전환할 준비가 되었는지에 대한 서버 응답을 확인하지 않습니다('Ready to st

//--- TLS 핸드셰이크 프로토콜을 사용하여 지정된 호스트에 대한 보안 TLS(SSL) 연결을 시작
   if(SocketTlsHandshake(socket,InpTimeout))
      return(true);

   Print("SocketTlsHandshake() failed. Error ",GetLastError());
   return(false);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 소켓을 생성하고 핸들을 수신
   int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- 포트를 통해 서버에 접속
   if(!SocketConnect(socket,SERVER,PORT,10000))
     {
      Print("SocketConnect() failed. Error ",GetLastError());
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 안전하지 않은 연결이 설정됨
      PrintFormat("%s connection has been established to %s:%d",(PORT==443 ? "A secure
      //--- 안전한 연결로 전환
      if(PORT!=443 && TlsHandshake(socket))
        {
         PrintFormat("Unsecured connection to %s:%d switched to secured",SERVER,PORT);
         //--- 연결이 인증서로 보호되는 경우 해당 데이터를 표시합니다
         string   subject,issuer,serial,thumbprint;
         datetime expiration;

         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:      ",subject);
            Print("   Issuer:     ",issuer);
            Print("   Number:     ",serial);
            Print("   Print:      ",thumbprint);
            Print("   Expiration: ",expiration);
           }
        }
     }
//--- 사용 후 소켓 종료
   SocketClose(socket);
   /*
  결과:
   An unsecured connection has been established to smtp.gmail.com:587
   SERVER: 220 smtp.gmail.com ESMTP a640c23a62f3a-a9b1f298319sm82305866b.105 - gsmtp

   SERVER: 250-smtp.gmail.com at your service, [37.193.40.122]
   250-SIZE 35882577
   250-8BITMIME
   250-STARTTLS
   250-ENHANCEDSTATUSCODES
   250-PIPELINING
   250-CHUNKING
   250 SMTPUTF8

   SERVER: 220 2.0.0 Ready to start TLS

   SocketTlsHandshake(): A secure connection to smtp.gmail.com:587 is now established
   TLS certificate:
      Owner:  /CN=smtp.gmail.com
      Issuer:  /C=US/O=Google Trust Services/CN=WR2
      Number:     1f:f4:db:2a:5a:e6:dc:52:0a:4c:05:ce:81:cc:c3:f7
      Print: d6be8af229b5329cd3d4c2789c02aa94f89b421c
      Expiration: 2024.12.30 08:25:30
   */
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 스트링을 서버에 전송                                                     |
//+------------------------------------------------------------------+
bool SendString(int socket,const string str)
  {
//--- 문자열을 문자 배열로 변환
   uchar data[];
   int   size=StringToCharArray(str,data,0,str.Length(),CP_UTF8);
//--- 데이터를 소켓에 전송
   ResetLastError();
   if(SocketSend(socket,data,size)==size)
      return(true);
//--- 데이터 전송 에러
   Print("Failed to send data to server. Error ",GetLastError());
   return false;
  }
//+------------------------------------------------------------------+
//| 서버에서 문자열 가져오기                                           |
//+------------------------------------------------------------------+
bool RecvString(int socket,string& result,uint timeout_ms=1000)
  {
//--- 데이터가 소켓에 나타날 때까지 기다립니다
   ulong wait_time_end=GetMicrosecondCount()+timeout_ms*1000;

   while(!SocketIsReadable(socket))
     {
      Sleep(10);
      //--- 데이터 대기 시간 초과 - 응답으로 NULL 반환
      if(wait_time_end<GetMicrosecondCount())
        {
         Print("ERROR: No response from server");
         return(false);
        }
     }
//--- 소켓에서 데이터 읽음
   uchar  data[128];
   uint   size=0;
   string resp=NULL;

   do
     {
      uchar b[1];
      int   n=SocketRead(socket,b,1,1000);

      if(n < 0)
         break;

      if(n)
        {
         data[size++]=b[0];
```

© 2000-2025, MetaQuotes Ltd.

```
         if(size==data.Size())
           {
            resp += CharArrayToString(data,0,data.Size(),CP_UTF8);
            size = 0;
           }
        }
      }
   while(SocketIsReadable(socket));
//--- 읽은 데이터를 문자열에 복사
   if(size)
      resp+=CharArrayToString(data,0,size,CP_UTF8);
//--- 문자열이 비어 있으면 에러임
   if(!resp.Length())
     {
      Print("ERROR: No response from server");
      return(false);
     }
//--- 문자열 반환
   result=resp;
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

SocketTlsCertificate

네트워크 연결을 보호하는 데 사용되는 인증서의 데이터를 가져옵니다.

```
bool  SocketTlsCertificate(
   int           socket,              // 소켓
   string&       subject,             // 인증서 소유자
   string&       issuer,              // 인증서 발급자
   string&       serial,              // 인증서 일련 번호
   string&       thumbprint,          // 인증서 출력
   datetime&     expiration           // 인증서 만료일
   );
```

Parameter

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 전달된 경우, 5270 에러(ERR_NETSOCKET_INVALIDHANDLE)가 _LastError에 작성됩니다.

제목

[in] 인증서 소유자. 제목 필드에 해당합니다.

issuer

[in] 인증서 발급자 이름. 발급자 필드에 해당합니다.

serial

[in] 인증서 일련 번호. 일련 번호 필드에 해당합니다.

thumbprint

[in] 인증서 출력. 전체 인증서 파일(발행자 서명을 포함한 모든 필드)의 SHA-1 해시에 해당합니다.

expiration

[in] datetime 형식의 인증서 만료일.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

SocketTlsHandshake를 사용하여 보안 연결을 설정한 후에만 인증서 데이터를 요청할 수 있습니다.

인증서 획득에 오류가 발생할 시, 5275 에러(ERR_NETSOCKET_NO_CERTIFICATE)가 _LastError에 기록됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "터미널 설정에서 허용된 주소 목록에 주소를 추가하여 예제가 작동하도록 합
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| 서버로 명령 전송                                                 |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 포트 443을 통해 보안 TLS 연결을 사용하는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 표준 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 응답을 읽습니다                                             |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- 소켓이 여전히 존재하지만 제한 시간보다 길지 않을 때까지 소켓에서 데이터를 읽습니다
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 연결의 보안 여부에 따라 다양한 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 반응을 분석합니다
         if(rsp_len>0)
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 출력합니다
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP 응답 헤더 수신:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- 핸들 체크
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든 것이 문제 없다면 연결합니다
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("다음에 대한 연결 설정 ",주소,":",포트);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 인증서로 연결이 보안된 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS 인증서:");
            Print("    소유자:  ",subject);
            Print("    발행인:  ",issuer);
            Print("    숫자:      ",serial);
            Print("    Print: ",thumbprint;
            Print("    만료: ",expiration;
            ExtTLS=true;
           }
         //--- GET 요청을 서버로 전송합니다
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET 요청 발송됨");
            //--- 반응을 읽습니다
            if(!HTTPRecv(socket,1000))
               Print("응답 가져오기 실패, error ",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
            }
          else
             Print("응답 가져오기 실패, error ",GetLastError());
         }
       else
         {
          Print("다음에 연결 ",주소,":",포트," 실패, error ",GetLastError());
         }
       //--- 이용 후 소켓을 닫습니다
       SocketClose(socket);
      }
    else
       Print("소켓 생성 실패, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

SocketTlsRead

보안 TLS 연결에서 데이터를 읽습니다.

```
int  SocketTlsRead(
   int           socket,              // 소켓
   uchar&        buffer[],            // 소켓에서 데이터를 읽기 위한 버퍼
   uint          buffer_maxlen        // 읽을 바이트 수
   );
```

Parameter

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못되 핸들이 _LastError로 전달될 시, 5270 에러(ERR_NETSOCKET_INVALIDHANDLE)가 활성화됩니다.

buffer

[out] 데이터를 읽을 수 있는 uchar 유형 배열에 대한 레퍼런스입니다. 동적 배열 크기가 읽기 바이트 수만큼 증가합니다. 배열의 크기는 INT_MAX (2147483647)을 초과할 수 없습니다.

buffer_maxlen

[in] *buffer[]* 배열에 읽을 바이트 수. 배열에 맞지 않는 데이터는 소켓에 남아 있습니다. 그것들은 다음 SocketTLSRead 호출을 통해 받을 수 있습니다. *buffer_maxlen*은 INT_MAX (2147483647)를 초과할 수 없습니다.

반환값

성공적이면 읽기 바이트 수를 반환합니다. 오류인 경우, -1이 반환됩니다.

참고

함수 실행시 시스템 소켓에 오류가 발생하면, SocketConnect을 통해 설정된 연결이 중단됩니다.

지정된 양의 데이터를 수신하거나 타임아웃이 (SocketTimeouts)에 도달할 때까지 함수가 실행됩니다.

데이터 읽기 오류가 발생시, 5273 에러(ERR_NETSOCKET_IO_ERROR)가 _LastError에 기록됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketExample.mq5 |
//|                        Copyright 2018, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "Copyright 2000-2024, MetaQuotes Ltd."
#property link        "https://www.mql5.com"
#property version     "1.00"
#property description "터미널 설정에서 허용된 주소 목록에 주소를 추가하여 예제가 작동하도록 합
#property script_show_inputs
```

© 2000-2025, MetaQuotes Ltd.

```
input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| 서버로 명령 전송                                                 |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 포트 443을 통해 보안 TLS 연결을 사용하는 경우
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- 표준 TCP 연결을 사용하는 경우
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 서버 응답을 읽습니다                                             |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- 소켓이 여전히 존재하지만 제한 시간보다 길지 않을 때까지 소켓에서 데이터를 읽습니다
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- 연결의 보안 여부에 따라 다양한 읽기 명령
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- 반응을 분석합니다
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- 응답 헤더만 출력합니다
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP 응답 헤더 수신:");
               Print(StringSubstr(result,0,header_end));
               return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
               }
            }
         }
      }
    while(GetTickCount()<timeout_check && !IsStopped());
    return(false);
   }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- 핸들 체크
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든 것이 문제 없다면 연결합니다
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("다음에 대한 연결 설정 ",주소,":",포트);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 인증서로 연결이 보안된 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS 인증서:");
            Print("    소유자:  ",subject);
            Print("    발행인:  ",issuer);
            Print("    숫자:     ",serial);
            Print("    Print: ",thumbprint;
            Print("    만료: ",expiration;
            ExtTLS=true;
           }
         //--- GET 요청을 서버로 전송합니다
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET 요청 발송됨");
            //--- 반응을 읽습니다
            if(!HTTPRecv(socket,1000))
               Print("응답 가져오기 실패, error ",GetLastError());
           }
         else
            Print("응답 가져오기 실패, error ",GetLastError());
        }
      else
        {
         Print("다음에 연결 ",주소,":",포트," 실패, error ",GetLastError());
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 이용 후 소켓을 닫습니다
      SocketClose(socket);
     }
   else
      Print("소켓 생성 실패, error ",GetLastError());
  }
//+------------------------------------------------------------------+
```

참고 항목

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketTlsReadAvailable

보안 TLS 연결에서 사용 가능한 모든 데이터를 읽습니다.

```
int  SocketTlsReadAvailable(
   int           socket,              // 소켓
   uchar&        buffer[],            // 소켓에서 데이터를 읽기 위한 버퍼
   const uint    buffer_maxlen        // 읽어들일 바이트 수
   );
```

매개 변수

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 _LastError로 전달될 시, 5270 에러(ERR_NETSOCKET_INVALIDHANDLE)가 활성화됩니다.

buffer

[out] 데이터를 읽을 수 있는 uchar 유형 배열에 대한 레퍼런스입니다. 동적 배열 크기가 읽기 바이트 수만큼 증가합니다. 배열의 크기는 INT_MAX (2147483647)을 초과할 수 없습니다.

buffer_maxlen

[in] buffer[] 배열에 읽을 바이트 수. 배열에 맞지 않는 데이터는 소켓에 남아 있습니다. 다음 SocketTlsReadAvailable 또는 SocketTlsRead 호출로 수신 가능. buffer_maxlen는 INT_MAX (2147483647)를 초과할 수 없습니다.

반환값

성공적이면 읽기 바이트 수를 반환합니다. 오류인 경우, -1이 반환됩니다.

참고

함수 실행시 시스템 소켓에 오류가 발생하면, SocketConnect을 통해 설정된 연결이 중단됩니다.

데이터 읽기 오류가 발생시, 5273 에러(ERR_NETSOCKET_IO_ERROR)가 _LastError에 기록됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                     SocketTlsReadAvailable.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property script_show_inputs
//+------------------------------------------------------------------+
//| Script inputs                                                    |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
input string InpMethod ="GET";             // Method (HEAD,GET)
input string InpServer ="www.google.com"; // Server
input uint   InpPort   =443;               // Port
input uint   InpTimeout=5000;              // Timeouts
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart(void)
  {
   Print("Server: ",InpServer);
   Print("Port: ",InpPort);
//--- 소켓을 생성하고 핸들을 수신
   const int socket=SocketCreate();

   if(socket==INVALID_HANDLE)
     {
      Print("SocketCreate() failed. Error ",GetLastError());
      return;
     }
//--- 소켓 시스템 객체와 관련한 데이터 수신 및 전송에 대한 시간 제한을 설정합니다.
   if(!SocketTimeouts(socket,InpTimeout,InpTimeout))
     {
      PrintFormat("SocketTimeouts(%u, %u) failed. Error %d",InpTimeout,InpTimeout,GetL
      SocketClose(socket);
      return;
     }
//--- 포트를 통해 서버에 접속
   if(!SocketConnect(socket,InpServer,InpPort,InpTimeout))
     {
      PrintFormat("SocketConnect('%s', %u, %u) failed. Error %d",InpServer,InpPort,Inp
      SocketClose(socket);
      return;
     }
//--- 네트워크 연결을 보호하는 데 사용되는 인증서에서 데이터를 가져옵니다
   string   subject,issuer,serial,thumbprint;
   datetime expiration;

   if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
     {
      Print("TLS certificate:");
      Print("   Owner:      ",subject);
      Print("   Issuer:     ",issuer);
      Print("   Number:     ",serial);
      Print("   Print:      ",thumbprint);
      Print("   Expiration: ",expiration);
     }
   else
     {
      //--- 서버가 인증서를 제공하지 않습니다 - 보안되지 않은 연결을 보고하고 종료
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("The connection is not secured by a certificate");
      SocketClose(socket);
      return;
     }
//--- 서버에 요청을 보냅니다
   string request=StringFormat("%s / HTTP/1.1\r\nHost: %s\r\nUser-Agent: MetaTrader 5\

   if(HTTPSendTLS(socket,request))
     {
      //--- 요청 전송됨 - 응답 받기
      Print("\nRequest sent. Starting page loading...");
      uchar response[]; //수신된 모든 데이터(문서 헤더 및 본문)

      if(!HTTPRecvTLS(socket,response,InpTimeout))
        {
         Print("There were errors while reading the page");
         SocketClose(socket);
         return;
        }
      //--- 수신된 데이터 바이트 수 보고
      PrintFormat("%u bytes received",response.Size());
      //--- 획득한 페이지 헤더만 표시
      string result    =CharArrayToString(response,0,WHOLE_ARRAY,CP_UTF8);
      int    header_end=StringFind(result,"\r\n\r\n");

      if(header_end>0)
        {
         Print("\nHTTP answer header received:");
         Print(StringSubstr(result,0,header_end));
        }
     }
//--- 사용 후 소켓 종료
   SocketClose(socket);
   /*
  결과:
   Server: www.google.com
   Port: 443
   TLS certificate:
      Owner: /CN=www.google.com
      Issuer: /C=US/O=Google Trust Services/CN=WR2
      Number: 0d:43:b1:4a:bb:9c:15:96:10:e1:3d:55:23:9f:25:4e
      Print: 89167618e5017f813aff981c88ce422dc1016bdf
      Expiration: 2024.12.30 08:26:35

   Request sent. Starting page loading...
   HTTPRecvTLS: Document received within 27 attempts
   25185 bytes received

   HTTP answer header received:
```

© 2000-2025, MetaQuotes Ltd.

```
   HTTP/1.1 200 OK
   Date: Fri, 25 Oct 2024 17:12:42 GMT
   Expires: -1
   Cache-Control: private, max-age=0
   Content-Type: text/html; charset=ISO-8859-1
   Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';script-src '
   Accept-CH: Sec-CH-Prefers-Color-Scheme
   P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
   Server: gws
   X-XSS-Protection: 0
   X-Frame-Options: SAMEORIGIN
   Set-Cookie: AEC=AVYB7coyYMCdweTDTaWeGYzmRnxzKGqsOEosH_VkbCn8xhWkFz6v0kxQFw; expires
   Set-Cookie: NID=518=J02X02Ff4v_9sMcNoUz-1SolmuG08E26Gs438ik0J_SOJUMy7of-P-qup-LaNSW
   Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
   Accept-Ranges: none
   Vary: Accept-Encoding
   Transfer-Encoding: chunked
   */
  }
//+------------------------------------------------------------------+
//| 보안 연결을 통해 HTTP 요청 보내기                                |
//+------------------------------------------------------------------+
bool HTTPSendTLS(int socket,const string request)
  {
//--- 문자열을 문자 배열로 변환하고 종료 0을 버립니다.
   char req[];
   int  len=StringToCharArray(request,req,0,WHOLE_ARRAY,CP_UTF8)-1;

   if(len<0)
      return false;

   return(SocketTlsSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| 보안 연결을 통해 웹페이지 가져오기                               |
//+------------------------------------------------------------------+
bool HTTPRecvTLS(int socket,uchar &response[],const uint timeout_ms)
  {
//--- 제한 시간이 만료되기 전에 보안 TLS 연결에서 사용 가능한 데이터를 읽습니다
   ulong timeout_check=GetTickCount64()+timeout_ms;
   uchar block[1024];   //소켓에서 읽는 블록 데이터를 위한 버퍼
   uint  attempt=0;     // 요청된 데이터 블록 수
   int   err    =0;     // 에러 코드

   ResetLastError();

   do
     {
      //--- 블록에서 일기 최대 1024 바이트
```

© 2000-2025, MetaQuotes Ltd.

```
      int len=SocketTlsReadAvailable(socket,block,1024);

      if(len>0)
        {
         attempt++;
         //--- 획득한 데이터 블록 합치기
         ArrayCopy(response,block,response.Size());
         //--- 획득한 데이터를 분석하고 헤더, 페이지 본문, 종료 또는 로딩 오류 등을 정의
         //...
         //...
         //...
         timeout_check=GetTickCount64()+timeout_ms;
        }
      else
         Sleep(10);

      err=GetLastError();
     }
   while(!IsStopped() && GetTickCount()<timeout_check && !err);
//--- 읽는 동안 오류가 있었나요?
   if(err)
     {
      Print("Error ",err);
      return(false);
     }

   PrintFormat("%s: Document received within %d attempts",__FUNCTION__,attempt);
   return(true);
  }
```

참고 항목

SocketTimeouts, MathSwap

© 2000-2025, MetaQuotes Ltd.

SocketTlsSend

보안 TLS 연결을 통해 데이터를 전송합니다.

```
int  SocketTlsSend(
   int           socket,              // 소켓
   const uchar&  buffer[],            // 데이터 버퍼
   uint          buffer_len           // 버퍼 사이즈
   );
```

매개 변수

socket

[in] 소켓 핸들이 SocketCreate 함수에 의해 반환됩니다. 잘못된 핸들이 전달된 경우, 5270 에러 (ERR_NETSOCKET_INVALIDHANDLE)가 _LastError에 작성됩니다.

buffer

[in] 전송할 데이터가 포함된 uchar 유형 배열에 대한 레퍼런스.

buffer_len

[in] '버퍼' 배열 사이즈.

반환값

성공적이면 소켓에 작성된 바이트 수를 반환합니다. 오류인 경우, -1이 반환됩니다.

참고

함수 실행시 시스템 소켓에 오류가 발생하면, SocketConnect을 통해 설정된 연결이 중단됩니다.

데이터 쓰기 오류가 발생시, 5273 에러(ERR_NETSOCKET_IO_ERROR)가 _LastError에 기록됩니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 호출하는 경우, GetLastError() 는 4014 에러-"호출이 허용되지 않는 함수입니다."가 반환됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                SocketTlsSend.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"
#property description "예제가 작동할 수 있도록 터미널 설정에서 허용된 주소 목록에 추가하세요"
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =443;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart(void)
  {
   int socket=SocketCreate();
//--- 핸들을 확인하세요
   if(socket!=INVALID_HANDLE)
     {
      //--- 모든게 준비되면 연결하세요
      if(SocketConnect(socket,Address,Port,1000))
        {
         PrintFormat("Established connection to %s:%d",Address,Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- 연결이 인증서로 보호되는 경우 해당 데이터를 표시합니다
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS certificate:");
            Print("   Owner:      ",subject);
            Print("   Issuer:     ",issuer);
            Print("   Number:     ",serial);
            Print("   Print:      ",thumbprint);
            Print("   Expiration: ",expiration);
           }
         //--- GET 요청을 서버에 보내세요
         string request="GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r\n\r
         char   req[];
         int    len=StringToCharArray(request,req)-1;

         if(len<0)
           {
            Print("StringToCharArray() failed. Error ", GetLastError());
            SocketClose(socket);
            return;
           }
         //--- 포트 443을 통한 보안 TLS 연결이 사용되는 경우
         if(SocketTlsSend(socket,req,len)==len)
           {
            Print("GET request sent");
            //--- 회신 읽기
            if(!HTTPRecv(socket,1000))
               Print("Failed to get a response, error ",GetLastError());
           }
         else
            Print("Failed to send GET request, error ",GetLastError());
        }
      else
        {
         PrintFormat("Connection to %s:%d failed, error %d",Address,Port,GetLastError
```

© 2000-2025, MetaQuotes Ltd.

```
      }
     //--- 사용 후 소켓 종료
     SocketClose(socket);
    }
  else
     Print("Failed to create a socket, error ",GetLastError());
 }
//+------------------------------------------------------------------+
//| 서버 회신 읽기                                                   |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout_ms)
 {
//--- 데이터가 있는 동안 소켓에서 데이터를 읽습니다. 그러나 시간 초과보다 길지 않도록 합니다.
  char   rsp[];
  string result;
  ulong  timeout_check=GetTickCount64()+timeout_ms;

  do
    {
     uint len=SocketIsReadable(socket);

     if(len)
       {
        //--- 보안 TLS 연결 데이터 읽기 및 구문 분석
        int rsp_len=SocketTlsRead(socket,rsp,len);

        if(rsp_len>0)
          {
           result+=CharArrayToString(rsp,0,rsp_len);
           //--- 응답 헤더만 표시
           int header_end=StringFind(result,"\r\n\r\n");

           if(header_end>0)
             {
              Print("HTTP answer header received:");
              Print(StringSubstr(result,0,header_end));
              return(true);
             }
           //--- 읽기 제한 시간 만료 시간 업데이트
           timeout_check=GetTickCount64()+timeout_ms;
          }
       }
    }
  while(GetTickCount64()<timeout_check && !IsStopped());

  return(false);
 }
```

참고 항목

© 2000-2025, MetaQuotes Ltd.

SocketTimeouts, MathSwap, StringToCharArray

© 2000–2025, MetaQuotes Ltd.

WebRequest

함수는 지정된 서버로 HTTP 요청을 보냅니다. 함수에는 두 가지 버전이 있습니다.

1. 헤더 Content-Type: application/x-www-form-urlencoded 를 사용해 "key=value" 유형의 단순 호출을 전송합니다.

```
int  WebRequest(
   const string      method,           // HTTP method
   const string      url,              // URL
   const string      cookie,           // cookie
   const string      referer,          // referer
   int               timeout,          // 타임아웃
   const char        &data[],          // HTTP 메시지 본문의 배열
   int               data_size,        // data[] 배열의 바이트 크기
   char              &result[],        // 서버 응답 데이터를 포함하는 배열
   string            &result_headers   // 서버 응답 헤더
   );
```

2. 다양한 웹 서비스와 보다 유연한 상호 작용을 위해 커스텀 헤더 집합을 지정하는 모든 유형의 요청을 보냅니다.

```
int  WebRequest(
   const string      method,           // HTTP method
   const string      url,              // URL
   const string      headers,          // headers
   int               timeout,          // 타임아웃
   const char        &data[],          // HTTP 메시지 본문의 배열
   char              &result[],        // 서버 응답 데이터를 포함하는 배열
   string            &result_headers   // 서버 응답 헤더
   );
```

Parameter

method

[in] HTTP 메소드.

url

[in] URL.

headers

[in] 줄 바꿈 "\r\n"으로 구분된 "key: value" 유형의 헤더 요청.

cookie

[in] 쿠키 값.

referer

[in] HTTP 요청의 레퍼러 헤더 값.

timeout

[in] 밀리세컨드 단위 타임아웃.

© 2000-2025, MetaQuotes Ltd.

data[]

[in] HTTP メッセージ本文の データ 配列.

data_size

[in] data[] 配列の サイズ.

result[]

[out] サーバ 応答 データを 含む 配列.

result_headers

[out] サーバ 反応 ヘッダ.

반환값

HTTP 서버 응답 코드 또는 오류에대해 -1.

참고

WebRequest() 기능을 사용하려면, "Options" 창의 "Expert Advisors" 탭에 있는 허용된 URL 목록에 필요한 서버의 주소를 추가하십시오. 서버 포트는 지정된 프로토콜에 따라 자동으로 선택됩니다. - "http://"의 경우 80, "https://"의 경우 443.

WebRequest() 함수는 동기식이며, 이는 프로그램이 프로그램 실행을 중단하고 요청한 서버의 응답을 기다린다는 것을 의미합니다. 응답 수신 지연이 클 수 있으므로, 지표 모든 지표와 차트가 하나의 심볼에 공유하는 공통 스레드에서 실행되므로 지표의 호출에 대해 함수를 사용할 수 없습니다. 심볼 차트 중 하나에 대한 지표 성능 지연으로 인해 동일한 심볼을 가진 모든 차트의 업데이트가 중지될 수 있습니다.

이 함수는 자체 실행 스레드에서 실행되므로 Expert Advisor 및 스크립트에서만 호출할 수 있습니다. 지표에서 함수를 호출하려고 하면, GetLastError()가 오류 4014 - "호출을 허용하지 않는 함수입니다"를 반환합니다.

WebRequest()는 Strategy Tester에서 실행될 수 없습니다.

예:

```
void OnStart()
  {
   string cookie=NULL,headers;
   char   post[],result[];
   string url="https://finance.yahoo.com";
//--- 서버에 대한 액세스를 설정하려면 URL "https://finance.yahoo.com"을 추가해야 합니다.
//--- 허용된 URL 목록(메인 메뉴->도구->옵션, "Expert Advisors" 탭):
//--- 마지막 오류 코드 재설정
   ResetLastError();
//--- Yahoo Finance에서 html 페이지 다운로드
   int res=WebRequest("GET",url,cookie,NULL,500,post,0,result,headers);
   if(res==-1)
     {
      Print("Error in WebRequest. Error code  =",GetLastError());
      //--- URL이 나열되지 않은 것 같습니다. 주소를 추가해야 한다는 메시지를 표시합니다.
      MessageBox("다음 주소를 추가하십시오 '"+url+"' 을 'Expert Advisors' 탭의 허용된 URL 목
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      if(res==200)
        {
         //--- 다운로드 성공
         PrintFormat("파일이 성공적으로 다운로드 되었음, 파일 크기 %d byte.",ArraySize(resu
         //PrintFormat("Server headers: %s",headers);
         //--- 데이터를 파일에 저장
         int filehandle=FileOpen("url.htm",FILE_WRITE|FILE_BIN);
         if(filehandle!=INVALID_HANDLE)
           {
            //--- result[] 배열의 내용을 파일에 저장
            FileWriteArray(filehandle,result,0,ArraySize(result));
            //--- 파일 닫기
            FileClose(filehandle);
           }
         else
            Print("FileOpen 에러. Error code =",GetLastError());
        }
      else
         PrintFormat("다운로드 '%s' 실패, 에러 코드 %d",url,res);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SendFTP

FTP 탭의 설정 창에 지정된 주소로 파일을 보냅니다.

```
bool  SendFTP(
   string  filename,           // ftp로 보낼 파일
   string  ftp_path=NULL       // ftp 카탈로그
   );
```

매개 변수

filename

[in] 보낸 파일의 이름.

ftp_path=NULL

[in] FTP 카탈로그. 디렉토리를 입력하지 않으면 설정에 명시된 디렉토리가 사용됩니다.

반환값

오류가 발생하면 'false'를 반환합니다.

참고

발송한 파일은 *terminal_directory\MQL5\files* 또는 그 하위 폴더에 있어야 합니다. 설정에 FTP 주소 및/또는 액세스 암호가 지정되지 않은 경우 전송이 수행되지 않습니다.

SendFTP() 함수는 Strategy Tester에서는 작동하지 않습니다.

예:

```
//+------------------------------------------------------------------+
//|                                                      SendFTP.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   FILENAME  "SomeFile.bin"
#define   PATH      NULL
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 터미널에서 FTP 주소로 파일을 보낼 수 있는 권한을 확인합니다.
   if(!TerminalInfoInteger(TERMINAL_FTP_ENABLED))
     {
      Print("Error. 클라이언트 터미널에 FTP 주소로 메시지를 보낼 수 있는 권한이 없습니다.");
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 파일 보내기
   ResetLastError();
   if(!SendFTP(FILENAME, PATH))
      Print("SendFTP() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

SendMail

이메일 탭의 설정 창에 지정된 주소로 이메일을 보냅니다.

```
bool  SendMail(
   string  subject,        // 헤더
   string  some_text       // 이메일 텍스트
   );
```

매개 변수

제목

[in] 이메일 헤더.

some_text

[in] 이메일 본문.

반환값

이메일이 송신 대기열에 포함된 경우 true이며, 그렇지 않은 경우엔 false입니다.

참고

설정에서 발송을 금지할 수 있으며, 이메일 주소도 생략할 수 있습니다. 에러 정보에 대해서는 GetLastError()를 호출하십시오.

SendMail() 함수는 Strategy Tester에서 작동하지 않습니다.

예:

```
//+------------------------------------------------------------------+
//|                                                     SendMail.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   SUBJECT   "Test SendMail"
#define   TEXT      "Text for SendMail() function"
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 터미널에서 이메일 전송 권한을 확인하세요
   if(!TerminalInfoInteger(TERMINAL_EMAIL_ENABLED))
     {
      Print("Error. 클라이언트 터미널이 이메일 메시지를 보낼 수 있는 권한이 없습니다.");
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 메일 보내기
   ResetLastError();
   if(!SendMail(SUBJECT, TEXT))
      Print("SendMail() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

SendNotification

"알림" 탭에서 MetaQuotes ID가 지정된 모바일 터미널로 푸시 알림을 보냅니다.

```
bool  SendNotification(
   string  text          // 알림 텍스트
   );
```

매개 변수

텍스트

[in] 알림 텍스트. 메시지 길이는 255자를 초과할 수 없습니다.

반환값

터미널에서 알림이 성공적으로 전송된 경우 true이고, 오류가 발생한 경우 false를 반환합니다. 알림 푸시 실패후 확인시, GetLastError ()는 다음 오류 중 하나를 반환할 수 있습니다

- 4515 - ERR_NOTIFICATION_SEND_FAILED,
- 4516 - ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 - ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 - ERR_NOTIFICATION_TOO_FREQUENT.

참고

SendNotification() 함수에는 초당 2회 이하의 요청과 분당 10회 이하의 요청이라는 엄격한 사용 제한이 설정되어 있습니다. 사용 빈도 모니터링은 다이내믹합니다. 제한 위반 시 함수가 비활성화될 수 있습니다.

SendNotification() 함수는 Strategy Tester에서 작동하지 않습니다.

예:

```
//+------------------------------------------------------------------+
//|                                             SendNotification.mq5 |
//|                                Copyright 2024, MetaQuotes Ltd.   |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com
#property version     "1.00"

#define   MESSAGE   "Test Message"
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- 터미널에서 알림을 보낼 수 있는 권한을 확인합니다.
   if(!TerminalInfoInteger(TERMINAL_NOTIFICATIONS_ENABLED))
     {
      Print("Error. 클라이언트 터미널이 알림을 보낼 수 있는 권한이 없습니다.");
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 알림 보내기
   ResetLastError();
   if(!SendNotification(MESSAGE))
      Print("SendNotification() failed. Error ",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

클라이언트 터미널의 글로벌 변수

글로벌 변수 작업을 위한 그룹 집합 기능이 있습니다.

클라이언트 터미널의 글로벌 변수를 mql5 프로그램의 글로벌 범위에서 언급된 변수와 혼동해서는 안됩니다.

글로벌 변수는 마지막 액세스 이후 4주 동안 클라이언트 터미널에 보관된 후 자동으로 삭제됩니다. 글로벌 변수에 대한 액세스는 새 값을 설정하는 것뿐만 아니라 글로벌 변수 값을 읽는 것도 가능합니다.

클라이언트 터미널에서 시작된 모든 mql5 프로그램에서 클라이언트 터미널의 글로벌 변수에 동시에 액세스할 수 있습니다.

기능	액션
GlobalVariableCheck	지정된 이름의 글로벌 변수가 있는지 확인
GlobalVariableTime	글로벌 변수에 마지막으로 액세스한 시간을 반환
GlobalVariableDel	글로벌 변수 삭제
GlobalVariableGet	글로벌 변수 값 반환
GlobalVariableName	글로벌 변수 목록에서 글로벌 변수의 이름을 순서 번호별로 반환
GlobalVariableSet	새 값을 글로벌 변수로 설정
GlobalVariablesFlush	모든 글로벌 변수의 내용을 디스크에 강제 저장
GlobalVariableTemp	터미널의 현재 세션에만 존재하는 글로벌 변수로 새 값을 설정
GlobalVariableSetOnCondition	조건별로 기존 글로벌 변수의 새 값을 설정
GlobalVariablesDeleteAll	이름에 지정된 접두사가 있는 글로벌 변수를 삭제
GlobalVariablesTotal	글로벌 변수의 전체 수를 반환

© 2000-2025, MetaQuotes Ltd.

GlobalVariableCheck

지정된 이름의 글로벌 변수가 있는지 확인

```
bool  GlobalVariableCheck(
   string  name      // 글로벌 변수 이름
   );
```

매개변수

이름

[in] 글로벌 변수 이름.

값 반환

글로벌 변수가 있으면 true를, 그렇지 않으면 false를 반환합니다.

글로벌 변수는 마지막 사용 후 4주 동안 클라이언트 터미널에 존재하면 자동으로 삭제됩니다.

추가 참조

GlobalVariableTime()

© 2000-2025, MetaQuotes Ltd.

GlobalVariableTime

글로벌 변수에 마지막으로 액세스한 시간을 반환.

```
datetime  GlobalVariableTime(
   string  name      // 이름
   );
```

매개변수

이름

[in] 글로벌 변수 이름.

값 반환

함수는 지정된 글로벌 변수에 마지막으로 액세스한 시간을 반환합니다. GlobalVariableGet() 및 GlobalVariableCheck() 기능을 사용하여 변수에 주소를 지정하면 마지막 액세스 시간도 수정됩니다. 오류에 대한 자세한 내용을 얻기 위해 GetLastError() 기능을 호출하십시오.

주의

글로벌 변수는 마지막 호출 후 4주 동안 클라이언트 너미널에 존재합니다. 그 이후 자동으로 삭제됩니다.

추가 참조

GlobalVariableCheck()

© 2000-2025, MetaQuotes Ltd.

GlobalVariableDel

클라이언트 터미널에서 글로벌 변수를 삭제.

```
bool  GlobalVariableDel(
   string  name      // 글로벌 변수 이름
   );
```

매개변수

이름

[in] 글로벌 변수 이름.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError()함수를 호출해야 합니다.

주의

글로벌 변수는 마지막 사용 후 4주 동안 클라이언트 터미널에 존재하면 자동으로 삭제됩니다.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableGet

클라이언트 터미널의 기존 글로벌 변수 값을 반환. 함수의 2가지 변수가 있습니다.

1. 글로벌 변수의 값을 즉시 반환.

```
double  GlobalVariableGet(
   string  name      // 글로벌 변수 이름
   );
```

2. 함수 실행의 성공 여부에 따라 true 또는 false를 반환합니다. 성공하면 클라이언트 터미널의 글로벌 변수가 두 번째 매개변수에서 참조로 전달된 변수에 배치됩니다.

```
bool  GlobalVariableGet(
   string  name,              // 글로벌 변수 이름
   double& double_var         // 이 변수에는 글로벌 변수의 값이 포함됩니다
   );
```

매개변수

이름

[in] 글로벌 변수 이름.

double_var

[out] 클라이언트 터미널의 글로벌 변수에 저장된 값을 수락하는 double 유형의 대상 변수입니다.

값 반환

오류가 발생한 경우 기존 글로벌 변수의 값 또는 0. 오류에 대한 자세한 내용은 GetLastError()호출하십시오.

주의

글로벌 변수는 마지막 사용 후 4주 동안 클라이언트 터미널에 존재하면 자동으로 삭제됩니다.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableName

글로벌 변수의 이름을 순서의 번호로 반환.

```
string  GlobalVariableName(
   int  index      // 글로벌 변수 목록의 글로벌 변수 번호
   );
```

매개변수

인덱스

[in] 글로벌 변수 목록의 시퀀스 번호. 이 값은 0보다 크거나 같아야 하며 GlobalVariablesTotal()보다 작아야 합니다.

값 반환

글로벌 변수 목록에서 순서 번호에 따른 글로벌 변수 이름. 오류에 대한 자세한 내용은, GetLastError()를 호출하십시오.

주의

글로벌 변수는 마지막 사용 후 4주 동안 클라이언트 터미널에 존재하면 자동으로 삭제됩니다.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableSet

글로벌 변수의 새 값을 설정. 변수가 없으면 시스템에서 새 글로벌 변수를 생성합니다.

```
datetime  GlobalVariableSet(
   string  name,      // 글로벌 변수 이름
   double  value      // 설정할 값
   );
```

매개변수

이름

[in] 글로벌 변수 이름.

값

[in] 새 숫자 값.

값 반환

성공하면 함수는 마지막 수정 시간을, 그렇지 않으면 0을 반환합니다. 오류에 대한 자세한 내용은, GetLastError()를 호출하십시오.

주의

글로벌 변수 이름은 63자를 초과할 수 없습니다. 글로벌 변수는 마지막 사용 후 4주 동안 클라이언트 터미널에 존재하면 자동으로 삭제됩니다.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesFlush

모든 글로벌 변수의 내용을 디스크에 강제 저장.

```
void  GlobalVariablesFlush();
```

값 반환

반환 값 없음.

주의

터미널은 작업이 완료되면 모든 글로벌 변수를 기록하지만 갑작스러운 컴퓨터 작동 오류 시 데이터가 손실될 수 있습니다. 이 함수는 만일의 경우에 글로벌 변수를 저장하는 프로세스를 독립적으로 제어할 수 있도록 합니다.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableTemp

함수는 임지 글로벌 변수를 생성하려고 시도합니다. 변수가 없으면 시스템에서 새로운 임시 글로벌 변수를 생성합니다.

```
bool  GlobalVariableTemp(
   string  name      // 글로벌 변수 이름
   );
```

매개변수

이름

[in] 임시 글로벌 변수의 이름.

값 반환

성공하면 함수는 true를, 그렇지 않은 경우 false를 반환합니다. 오류에 대한 자세한 정보를 얻으려면 GetLastError() 기능을 호출하십시오.

주의

임시 글로벌 변수는 클라이언트 터미널이 실행 중일 때만 존재하며 터미널 종료 후 자동으로 삭제됩니다. GlobalVariablesFlush()를 실행하는 동안 임시 글로벌 변수가 디스크에 기록되지 않습니다.

임시 글로벌 변수가 생성되면 클라이언트 터미널의 글로벌 변수와 동일하게 액세스 및 수정할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

GlobalVariableSetOnCondition

현재 값이 세 번째 매개변수 check_value와 같을 경우 기존 글로벌 변수의 새 값을 설정합니다. 글로벌 변수가 없으면 함수는 ERR_GLOBALVARIABLE_NOT_FOUND (4501) 오류를 생성하고 false를 반환합니다.

```
bool  GlobalVariableSetOnCondition(
   string  name,            // 글로벌 변수 이름
   double  value,           // 조건이 true인 경우 변수에 대한 새 값
   double  check_value      // 값 조건 조회
   );
```

매개변수

이름

[in] 글로벌 변수 이름.

값

[in] 새 값.

check_value

[in] 글로벌 변수의 현재 값을 확인하는 값.

값 반환

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용은 GetLastError()를 호출하십시오. 글로벌 변수의 현재 값이 check_value와 다르면 함수는 false를 반환합니다.

주의

함수는 글로벌 변수에 대한 아토믹 액세스를 제공하므로 하나의 클라이언트 터미널에서 동시에 작동하는 엑스퍼트 어드바이저의 상호 작용에 따라 뮤텍스를 제공하는 데 사용할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesDeleteAll

클라이언트 터미널의 글로벌 변수를 삭제.

```
int  GlobalVariablesDeleteAll(
   string     prefix_name=NULL,     // 접두사로 시작하는 이름이 포함된 모든 글로벌 변수
   datetime   limit_data=0          // 이 날짜 이전에 변경된 모든 글로벌 변수
   );
```

매개변수

prefix_name=NULL

[in] 제거할 접두사 글로벌 변수 이름 지정. 접두사 NULL 또는 빈 문자열을 지정하면 데이터 조건을 충족하는 모든 변수가 삭제됩니다.

limit_data=0

[in] 글로벌 변수를 마지막으로 수정한 시점으로 선택할 날짜. 함수는 이 날짜 이전에 변경된 전역 변수를 제거합니다. 매개변수가 0이면 첫 번째 기준(접두사)을 충족하는 모든 변수가 삭제됩니다.

값 반환

삭제된 변수의 수.

주의

두 옵션이 모두 0(prefix_name = NULL 및 limit_data = 0)이면 함수는 터미널의 모든 글로벌 변수를 삭제합니다. 두 매개변수를 모두 지정하면 두 매개변수에 해당하는 글로벌 변수가 삭제됩니다.

글로벌 변수는 마지막 사용 후 4주 동안 클라이언트 터미널에 존재하면 자동으로 삭제됩니다.

© 2000-2025, MetaQuotes Ltd.

GlobalVariablesTotal

클라이언트 터미널의 총 글로벌 변수 수를 반환.

```
int  GlobalVariablesTotal();
```

값 반환

글로벌 변수의 수.

주의

글로벌 변수는 마지막 사용 후 4주 동안 클라이언트 터미널에 존재하면 자동으로 삭제됩니다. 글로벌 변수를 호출하면 새 값을 설정할 뿐만 아니라 글로벌 변수의 값을 읽을 수도 있습니다.

© 2000-2025, MetaQuotes Ltd.

파일 기능

파일 작업을 위한 기능 그룹입니다.

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

작업 파일을 찾을 수 있는 디렉토리는 두 개(하위 디렉토리 포함)입니다:

- terminal_data_folder₩MQL5₩FILES₩(터미널 메뉴에서 "파일" -"데이터 디렉토리 열기"를 선택);
- 컴퓨터에 설치된 모든 터미널의 공통 폴더 - 일반적으로 C:₩Documents and Settings₩All Users₩Application Data₩MetaQuotes₩Terminal₩Common₩Files 디렉토리에 위치.

TerminalInfoString() 함수를 사용하여 ENUM_TERMINAL_INFO_STRING 열거를 사용하여 이러한 카탈로그의 이름을 가져오는 프로그램 메서드가 있습니다:

```
//--- 터미널 데이터를 저장하는 폴더
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- 모든 클라이언트 터미널의 공통 폴더
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

다른 디렉토리의 파일을 사용하는 것은 금지됨.

파일 기능을 통해 소위 "이름 지정 파이프"로 작업할 수 있습니다. 이를 위하여, 적절한 매개변수를 사용하여 FileOpen() 기능을 호출하기만 하면 됩니다.

기능	액션
FileSelectDialog	파일 또는 폴더 열기/대화상자 생성
FileFindFirst	지정된 필터에 따라 디렉토리에서 파일 검색을 시작
FileFindNext	FileFindFirst() 기능으로 시작한 검색을 계속 진행
FileFindClose	검색 핸들을 닫기
FileOpen	지정된 이름과 플래그가 있는 파일을 열기
FileDelete	지정된 파일을 삭제
FileFlush	입력/출력 파일 버퍼에 남아있는 모든 데이터를 디스크에 작성
FileGetInteger	파일의 정수 속성을 가져오기
FileIsEnding	읽는 과정에서 파일의 끝을 정하기
FileIsLineEnding	읽는 과정 중 텍스트 파일에서 줄의 끝을 정하기
FileClose	이전에 연 파일을 닫기
FileIsExist	파일이 있는지 확인
FileCopy	로컬 또는 공유 폴더에서 다른 파일로 원본 파일 복사
FileMove	파일 이동 또는 이름 변경
FileReadArray	BIN 유형의 파일에서 문자열을 제외한 모든 유형의 배열을 읽기

© 2000-2025, MetaQuotes Ltd.

기능	액션
FileReadBool	CSV의 파일에서 현재 위치에서 구분 기호(또는 텍스트 줄의 끝)까지 문자열을 입력하고읽고 문자열을 bool 유형 갑으로 변환
FileReadDatetime	CSV 파일에서 다음 형식중 하나의 문자열을 읽습니다: "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" 또는 "HH:MM:SS" - 이 값을 날짜시간 값으로 변환
FileReadDouble	파일 포인터의 현재 포지션에서 이중 값을 읽기
FileReadFloat	파일 포인터의 현재 포지션에서 플로트(float) 값을 읽기
FileReadInteger	파일 포인터의 현재 위치에서 int, short 또는 char 값을 읽기
FileReadLong	파일 포인터의 현재 포지션에서 long 유형 값 읽기
FileReadNumber	CSV 파일의 현재 포지션에서 구분 기호(또는 텍스트 줄의 끝)까지 문자열을 읽기 및 문자열을 이중값으로 변환
FileReadString	파일에서 파일 포인터의 현재 포지션의 문자열 읽기
FileReadStruct	바이너리 파일의 내용을 파일 포인터의 현재 포지션에서 매개변수로 전달된 구조로 읽기
FileSeek	파일 포인터의 포지션을 지정된 바이트 수만큼 지정된 포지션으로 이동
FileSize	해당 열기 파일의 크기를 반환
FileTell	해당 열기 파일의 파일 포인터의 현재 포지션을 반환
FileWrite	CSV 또는 TXT 유형의 파일에 데이터 쓰기
FileWriteArray	문자열을 제외한 모든 유형의 배열을 BIN 유형 파일에 쓰기
FileWriteDouble	파일 포인터의 현재 포지션에서 바이너리 파일에 double 유형의 값을 쓰기
FileWriteFloat	파일 포인터의 현재 포지션에서 바이너리 파일에 float 유형의 값 쓰기
FileWriteInteger	파일 포인터의 현재 포지션에서 바이너리 파일에 int 유형의 값 쓰기
FileWriteLong	파일 포인터의 현재 포지션에서 바이너리 파일에 long 유형의 값 쓰기
FileWriteString	파일 포인터의 현재 포지션에서 시작하여 BIN 또는 TXT 파일에 문자열 매개변수 값을 쓰기
FileWriteStruct	파일 포인터에서 현재 포지션에서 시작하여 매개변수로 전달된 구조의 내용을 바이너리 파일에 쓰기
FileLoad	지정된 바이너리 파일의 모든 데이터를 전달된 일련의 숫자 유형 또는 단순 구조로 읽기
FileSave	매개변수로 전달된 배열의 모든 요소를 바이너리 파일에 쓰기
FolderCreate	Files 디렉토리에 폴더를 생성
FolderDelete	선택된 디렉토리를 제거 폴더가 비어있지 않으면 제거할 수 없습니다.

© 2000-2025, MetaQuotes Ltd.

기능	액션
FolderClean	지정된 폴더의 모든 파일을 삭제

파일이FileOpen()을 사용하여 쓰기 위해 열려 있는 경우 경로에 지정된 모든 하위 폴더가 없으면 하위 폴더가 생성됩니다.

© 2000-2025, MetaQuotes Ltd.

FileSelectDialog

파일 또는 폴더 오프닝/생성 다이얼로그 생성.

```
int  FileSelectDialog(
   string   caption,              // 윈도우 헤더
   string   initial_dir,          // 초기 디렉토리
   string   filter,               // 확장 필터
   uint     flags,                // 플래그 조합
   string&  filenames[],          // 파일 이름이 있는 배열
   string   default_filename      // 디폴트 파일명
   );
```

Parameters

caption

[in] 대화창 헤더.

initial_dir

[in] MQL5₩Files에 상대적인 초기 디렉터리 이름, 내용은 대화 상자에 표시됨. 값이 NULL이면, MQL5₩Files 가 대화상자에 표시됩니다.

filter

[in] 선택 대화 상자에 표시할 파일의 확장자 필터. 다른 형식의 파일은 숨겨집니다.

flags

[in] 대화 창 모드를 정의하는 플래그 조합. 플래그는 다음과 같이 정의됩니다:
FSD_WRITE_FILE – 파일 오픈 다이얼로그;
FSD_SELECT_FOLDER – 폴더 선택만 허용;
FSD_ALLOW_MULTISELECT – 여러 파일 선택 허용;
FSD_FILE_MUST_EXIST – 선택한 파일이 있어야 함;
FSD_COMMON_FOLDER – 파일은 모든 클라이언트 터미널의 공통 폴더 terminals ₩Terminal₩Common₩Files에 있습니다.

filenames[]

[out] 선택한 파일/폴더의 이름이 배치되는 문자열 배열.

default_filename

[in] 디폴트 파일/폴더명. 지정된 경우 열린 대화 상자에 이름이 자동으로 추가되고 테스트 시 *filenames[]* 배열에 반환됩니다.

반환 값

성공적으로 완료되면 *filenames[]*에서 이름을 얻을 수 있는 선택한 파일의 수를 반환합니다. 사용자가 파일을 선택하지 않고 대화상자를 닫으면 함수는 0을 반환합니다. 실행에 실패하면, 0보다 작은 값이 반환됩니다. 오류 코드는 GetLastError()을 사용하여 획득할 수 있습니다.

참고

보안을 위해 MQL5 언어로 파일 작업을 엄격하게 제어합니다. MQL5 언어로 파일 작업에 사용되는 파일은 파일 샌드박스 외부(즉, MQL5₩Files 디렉터리 외부)에 있을 수 없습니다.

© 2000-2025, MetaQuotes Ltd.

initial_dir 이름은 클라이언트 터미널의 디렉터리에서 MQL5₩Files(또는 testing_agent_directory₩MQL5₩Files₩로 검색됩니다. If FSD_COMMON_FOLDER is set among the flags, the search for the initial directory is performed in the common folder of all client terminals \Terminal\Common\Files.

filter 파라미터는 유효한 파일을 나타내므로 다음과 같이 설정해야 합니다 "<description 1>|<extension 1>|<description 2>|<extension 2>..." 형태, 예를 들면, "Text files (*.txt)|*.txt|All files (*.*)|*.*". 첫 번째 확장자 "텍스트 파일(*.txt)|*.txt"가 기본 파일 형식으로 선택됩니다.

filter=NULL 일 경우, 대화창에서 파일 선택 마스크는 다음과 같습니다 "All Files (*.*)|*.*|"

default_filename 파라미터를 설정한 경우, FileSelectDialog()를 호출하면 1이 반환되며, 시각화되지 않은 테스트 중에 *default_filename* 값 자체가 *filenames[]* 배열에 복제됩니다.

FileSelectDialog()를 호출하면 사용자의 응답을 기다리는 동안 내내 실행 스레드 가 일시 중단되므로 커스텀 지표에 이 기능을 사용할 수 없습니다. 각 심볼에 대한 모든 지표가 단일 스레드로 실행되기 때문에 이러한 보류는 이 심볼에 대한 모든 시간대의 모든 차트를 작동할 수 없게 합니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 클라이언트 터미널의 공통 폴더에서 다운로드할 텍스트 파일의 이름을 가져옵니다.
   string filenames[];
   if(FileSelectDialog("다운로드할 파일을 선택하세요", NULL,
                       "Text files (*.txt)|*.txt|All files (*.*)|*.*",
                       FSD_ALLOW_MULTISELECT|FSD_COMMON_FOLDER, filenames, "data.txt")
     {
      //--- 선택한 각 파일의 이름 표시
      int total=ArraySize(filenames);
      for(int i=0; i<total; i++)
         Print(i, ": ", filenames[i]);
     }
   else
     {
     Print("파일이 선택되지 않았습니다");
     }
//---
  }
```

더 보기

FileOpen, FileIsExist, FileDelete, FileMove, FolderCreate, FolderDelete, FolderClean, Flags of opening files

© 2000-2025, MetaQuotes Ltd.

FileFindFirst

이 함수는 지정된 필터에 따라 디렉토리의 파일 또는 subdirectories 검색을 시작합니다.

```
long  FileFindFirst(
   const string   file_filter,          // 문자열 - 검색 필터
   string&        returned_filename,    // 발견된 파일 또는 하위 디렉토리의 이름
   int            common_flag=0         // 검색 정의
   );
```

Parameters

file_filter

[in] 검색 필터. 파일을 검색할 ₩files 디렉토리와 관련된 하위 디렉토리(또는 중첩된 하위 디렉토리 순서)를 필터에 지정할 수 있습니다.

returned_filename

[out] 반환된 매개 변수로, 이 매개 변수가 성공시 처음 발견된 파일 또는 하위 디렉토리의 이름이 배치됩니다. 파일명(확장자 포함), 디렉토리 및 하위 디렉토리만 검색 필터에 지정되었는지 여부에 관계없이 포함되지 않습니다.

common_flag

[in] 파일 위치를 결정짓는 플래그. common_flag = FILE_COMMON 라면, 파일은 모든 클라이언트 터미널의 공유 폴더인 ₩Terminal₩Common₩Files에 있습니다. 그렇지 않으면, 파일은 로컬 폴더에 있습니다.

반환 값

검색된 개체의 핸들을 반환합니다. 이 핸들은 FileFindNext() 함수를 기준으로 파일과 하위 디렉토리를 추가로 정렬하는 데 사용하거나, 파일이 없을 때 INVALID_HANDLE 및 필터에 해당하는 하위 디렉토리를 정렬하는 데 사용됩니다(특히 디렉토리가 비어 있을 때). 검색 후 FileFindClose() 함수를 사용하여 핸들을 닫아야 합니다.

참고

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

예:

© 2000-2025, MetaQuotes Ltd.

```
//--- 스크립트를 시작할 때 입력 매개 변수 창 표시
#property script_show_inputs
//--- filter
input string InpFilter="Dir1\\*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   string int_dir="";
   int    i=1,pos=0,last_pos=-1;
//--- 마지막 백슬래시 찾기
   while(!IsStopped())
     {
      pos=StringFind(InpFilter,"\\",pos+1);
      if(pos>=0)
         last_pos=pos;
      else
         break;
     }
//--- the filter contains the folder name
   if(last_pos>=0)
      int_dir=StringSubstr(InpFilter,0,last_pos+1);
//--- 로컬 폴더의 루트에 검색 핸들 가져오기
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- FileFindFirst()가 성공적으로 실행되었는지 확인
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 루프에서 전달된 문자열이 파일 또는 디렉토리의 이름인지 확인하십시오
      do
        {
         ResetLastError();
         //--- 파일이면 함수는 true를 반환하고 디렉토리이면 오류 ERR_FILE_IS_DIRECTORY 를 반
         FileIsExist(int_dir+file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- 검색 핸들 닫기
      FileFindClose(search_handle);
     }
   else
      Print("Files not found!");
  }
```

더 보기

FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

FileFindNext

이 함수는 FileFindFirst()에 의해 시작된 검색을 계속 합니다.

```
bool  FileFindNext(
   long      search_handle,         // 검색 핸들
   string&   returned_filename      // 발견된 파일 또는 하위 디렉토리의 이름
   );
```

Parameters

search_handle

[in] 검색 핸들, FileFindFirst()로 검색.

returned_filename

[out] 발견된 다음 파일 또는 하위 디렉토리의 이름. 파일명(확장자 포함), 디렉토리 및 하위 디렉토리만 검색 필터에 지정되었는지 여부에 관계없이 포함되지 않습니다.

반환 값

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 필터
input string InpFilter="*";
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- 로컬 폴더의 루트에서 검색 핸들 수신
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- FileFindFirst() 함수가 성공적으로 실행되었는지 확인
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 전달된 문자열이 루프의 파일 또는 디렉토리 이름인지 확인
      do
        {
         ResetLastError();
         //--- 파일일 경우 함수는 true를 반환하고 디렉토리일 경우 함수는 오류 ERR_FILE_IS_DI
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Di
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- 검색 핸들 닫기
```

© 2000-2025, MetaQuotes Ltd.

```
      FileFindClose(search_handle);
     }
   else
      Print("파일을 찾을 수 없습니다!");
  }
```

더 보기

FileFindFirst, FileFindClose

© 2000-2025, MetaQuotes Ltd.

FileFindClose

이 함수는 검색 핸들을 닫습니다.

```
void  FileFindClose(
   long  search_handle      //  검색 핸들
   );
```

Parameters

search_handle

[in] 검색 핸들, FileFindFirst()로 검색.

반환 값

값이 반환되지 않음.

참고

시스템 리소스를 확보하려면 함수를 호출해야 합니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 필터
input string InpFilter="*";
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- 로컬 폴더의 루트에서 검색 핸들 수신
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- FileFindFirst() 함수가 성공적으로 실행되었는지 확인
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 전달된 문자열이 루프의 파일 또는 디렉토리 이름인지 확인
      do
        {
         ResetLastError();
         //--- 파일일 경우 함수는 true를 반환하고 디렉토리일 경우 함수에서 오류 5018을 생성합니
         FileIsExist(file_name);
         PrintFormat("%d : %s name = %s",i,GetLastError()==5018 ? "Directory" : "File"
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- 검색 핸들 닫기
      FileFindClose(search_handle);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      Print("파일을 찾을 수 없습니다!");
  }
```

더 보기

FileFindFirst, FileFindNext

© 2000-2025, MetaQuotes Ltd.

FileIsExist

파일의 존재 여부를 확인합니다.

```
bool  FileIsExist(
   const string  file_name,        // 파일명
   int           common_flag=0     // 탐색 영역
   );
```

Parameters

file_name

[in] 확인 중인 파일의 이름

common_flag=0

[in] 파일 위치를 결정짓는 플래그. common_flag = FILE_COMMON 라면, 파일은 모든 클라이언트 터미널의 공유 폴더인 ₩Terminal₩Common₩Files에 있습니다. 그렇지 않으면, 파일은 로컬 폴더에 있습니다.

반환 값

지정한 파일이 있는 경우 true를 반환합니다.

참고

선택한 파일이 하위 디렉토리로 밝혀질 수 있습니다. 이 경우 FileIsExist() 함수는 false를 반환하고 오류 5018은 _LastError 변수에 기록됩니다 -"이것은 파일이 아닌 디렉토리입니다"(FileFindFirst 함수에 대한 예 참조)

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

common_flag = FILE_COMMON이면 함수는 모든 클라이언트 터미널 \Terminal\Common\Files의 공유 폴더에서 파일을 찾고, 그렇지 않으면 로컬 폴더(테스트의 경우 MQL5₩Files or MQL5₩Tester₩Files)에서 파일을 찾습니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 오래된 파일의 날짜
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // 파일 이름 저장 변수
   string   filter="*.txt"; // 파일 검색을 위한 필터
   datetime create_date;    // 파일 생성일
   string   files[];        // 파일명 목록
   int      def_size=25;    // 기본 배열 규모
   int      size=0;         // 파일 수
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 배열용 메모리 할당
   ArrayResize(files,def_size);
//--- 로컬 폴더의 루트에서 검색 핸들을 수신
   long search_handle=FileFindFirst(filter,file_name);
//--- FileFindFirst()이 성공적으로 실행한 경우 검증
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 루프테어 파일 검색
      do
        {
         files[size]=file_name;
         //--- 배열 규모를 늘림
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- 에러값 리셋
         ResetLastError();
         //--- 파일 생성일 수령
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- 파일이 오래되었는지 검증
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s 파일 삭제됨!",file_name);
            //--- 오래된 파일 삭제
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- 검색 핸들 닫기
      FileFindClose(search_handle);
     }
   else
     {
      Print("파일을 찾을 수 없습니다!");
      return;
     }
//--- 남아 있는 파일 확인
   PrintFormat("결과:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s 파일 존재!",files[i]);
      else
         PrintFormat("%s 파일 삭제됨!",files[i]);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

더 보기

FileFindFirst

© 2000-2025, MetaQuotes Ltd.

FileOpen

이 함수는 지정된 이름과 플래그를 가진 파일을 엽니다.

```
int  FileOpen(
   string  file_name,           // 파일명
   int     open_flags,          // 플래그 조합
   short   delimiter='\t',      // 구분기호
   uint    codepage=CP_ACP      // 코드 페이지
   );
```

Parameters

file_name

[in] 파일 이름에는 하위 폴더가 포함될 수 있음. 파일을 쓰기 위해 연 경우 이러한 하위 폴더가 없으면 하위 폴더가 생성됩니다.

open_flags

[in] 파일의 작동 모드를 결정하는 플래그 조합. 플래그는 다음과 같이 정의됩니다:
FILE_READ 파일은 읽기를 위해 열립니다
FILE_WRITE 파일은 쓰기를 위해 열립니다
FILE_BIN 이진 읽기-쓰기 모드 (문자열에서 문자열로의 변환 없음)
csv 형식의 FILE_CSV 파일(기록된 모든 항목은 UNICODE 또는 ANSI 형식의 문자열로 변환되며 구분 기호로 구분됨)
FILE_TXT 단순 텍스트 파일(csv와 동일하지만 구분 기호는 고려되지 않음)
ANSI 유형의 FILE_ANSI 라인(싱글바이트 기호)
UNICODE 유형의 FILE_UNICODE 줄(더블바이트 문자)
FILE_SHARE_READ 여러 프로그램의 공유 읽기
FILE_SHARE_WRITE 여러 프로그램의 공유 쓰기
FILE_COMMON 모든 클라이언트 터미널의 공유 폴더에 있는 파일 위치 ₩Terminal₩Common₩Files

delimiter='\t'

[in] txt 또는 csv 파일의 구분자로 사용할 값. csv-파일 구분 기호를 지정하지 않으면 기본적으로 탭으로 설정됩니다. txt 파일 구분 기호를 지정하지 않으면 구분 기호가 사용되지 않습니다. 구분 기호가 0으로 설정되어 있으면 구분 기호가 사용되지 않습니다.

codepage=CP_ACP

[in] 코드 페이지 값. 가장 많이 사용되는 코드 페이지는 적절한 상수를 제공합니다.

반환 값

파일이 성공적으로 열리면 함수는 파일 핸들을 반환하고 이 핸들을 파일 데이터에 액세스하는 데 사용합니다. 실패한 경우 INVALID_HANDLE를 반환합니다.

참고

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

파일을 특정 인코딩으로 읽어야 하는 경우 FILE_ANSI 플래그를 설정해야 합니다 (코드 페이지 값이 지정된 코드 페이지 매개 변수가 지정됨). 지정된 FILE_ANSI 플래그가 없는 경우 텍스트 파일은 변환 없이 Unicode로 읽힙니다.

© 2000-2025, MetaQuotes Ltd.

파일이 하위 폴더 MQL5₩Files (또는 testing_agent_directory₩MQL5₩Files)에 있는 클라이언트 터미널 폴더에 열립니다. 플래그 사이에 FILE_COMMON가 지정된 경우 파일은 모든 MetaTrader 5 클라이언트 터미널의 공유 폴더에 열립니다.

명명된 파이프는 다음 규칙에 따라 열 수 있습니다:

- 파이프 이름은 다음과 같은 모습을 가져야 하는 문자열입니다: "₩₩servername₩pipe₩pipename", 여기서 servername은 네트워크 서버 이름이고 pipename은 파이프 이름입니다. 파이프가 동일한 시스템에서 사용되는 경우 서버 이름을 생략할 수 있지만 포인트 대신 "₩₩.₩pipe₩pipename"을 삽입해야 합니다. 파이프를 연결하려는 클라이언트는 파이프 이름을 알고 있어야 합니다.
- FileFlush() 및 FileSeek()는 파이프에서 읽고 쓰는 순차 작업 사이에 파일 시작 부분에 호출되어야 합니다.

특수 기호 '₩'는 표시된 문자열에 사용됩니다. 따라서 MQL5 애플리케이션에서 이름을 쓸 때는 '₩'를 두번씩 사용해야 합니다. 이는 위의 예제가 코드에서 "₩₩servername₩pipe₩pipename"으로 표시되어야 함을 의미합니다.

명명된 파이프로 작업하는 방법에 대한 자세한 내용은 "DLL을 사용하지 않고 명명된 파이프를 사용하여 MetaTrader 5와 통신" 문서를 참조하십시오.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 잘못된 파일 열기 방법
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
   int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
   if(filehandle<0)
     {
      Print("절대 경로로 파일을 열지 못했습니다. ");
      Print("오류 코드 ",GetLastError());
     }

//--- "파일 샌드박스"에서 올바른 작업 방법
   ResetLastError();
   filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
   if(filehandle!=INVALID_HANDLE)
     {
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("FileOpen OK");
     }
   else Print("FileOpen 작업 실패, 오류 ",GetLastError());
//--- MQL5₩Files₩에 폐쇄된 디렉토리를 생성하는 또 다른 예
   string subfolder="Research";
   filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
      if(filehandle!=INVALID_HANDLE)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));
      FileClose(filehandle);
      Print("파일은 다음 폴더에 생성해야 합니다"+terminal_data_path+"\\"+subfolder);
     }
   else Print("파일 열기 실패, 에러 ",GetLastError());
  }
```

더 보기

Use of a Codepage, FileFindFirst, FolderCreate, File opening flags

© 2000-2025, MetaQuotes Ltd.

FileClose

이전에 FileOpen()에서 연 파일을 닫습니다.

```
void  FileClose(
   int  file_handle      // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에 의해 반환된 파일 서술자.

반환 값

값이 반환되지 않음.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 입력 매개 변수
input string InpFileName="file.txt";     // 파일명
input string InpDirectoryName="Data";    // 디렉토리명
input int    InpEncodingType=FILE_ANSI; // ANSI=32 또는 UNICODE=64
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용할 파일의 경로를 인쇄
   PrintFormat("Working %s\\Files\\ folder",TerminalInfoString(TERMINAL_DATA_PATH));
//--- 에러값 리셋
   ResetLastError();
//--- 읽기 목적으로 파일 열기(파일이 없는 경우 오류 발생).
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //--- 파일 내용 인쇄
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- 파일 닫기
      FileClose(file_handle);
     }
   else
      PrintFormat("Error, code = %d",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

FileCopy

이 함수는 로컬 또는 공유 폴더에서 다른 파일로 원본 파일을 복사합니다.

```
bool  FileCopy(
   const string  src_file_name,     // 소스파일명
   int           common_flag,       // 위치
   const string  dst_file_name,     // 대상 파일명
   int           mode_flags         // 엑세스 모드
   );
```

Parameters

src_file_name

[in] 복제할 파일 명.

common_flag

[in] 파일 위치를 결정짓는 플래그. common_flag = FILE_COMMON 라면, 파일은 모든 클라이언트 터미널의 공유 폴더인 ₩Terminal₩Common₩Files에 있습니다. 그렇지 않으면, 파일은 로컬 폴더에 있습니다 (예, *common_flag=0*).

dst_file_name

[in] 결과 파일명.

mode_flags

[in] 엑세스 플래그. 매개 변수에는 FILE_REWRITE 및/또는 FILE_COMMON 2개의 플래그만 포함될 수 있습니다. 다른 플래그는 무시됩니다. 파일이 이미 존재하고 FILE_REW 플래그가 지정되지 않은 경우 파일이 다시 작성되지 않고 함수가 false를 반환합니다.

반환 값

오류가 발생하면 함수는 false를 반환합니다.

참고

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

새 파일이 이미 있는 경우 mode_flags 매개 변수에서 FILE_REWRITE 플래그가 사용 가능한지 여부에 따라 복제본이 만들어집니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 입력 매개 변수
input string InpSrc="source.txt";       // 소스
input string InpDst="destination.txt";  // 복제
input int    InpEncodingType=FILE_ANSI; // ANSI=32 또는 UNICODE=64
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 소스 내용을 표시 (존재해야 함)
   if(!FileDisplay(InpSrc))
      return;
//--- 복제 파일이 이미 있는지 확인 (생성할 수 없음)
   if(!FileDisplay(InpDst))
     {
      //--- 복제 파일이 존재하지 않습니다. FILE_REWRITE 플래그 없이 복사합니다(올바른 복제).
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("파일이 복제 되었습니다!");
      else
         Print("파일이 복제되지 않았습니다!");
     }
   else
     {
      //--- 복제 파일이 이미 있습니다. FILE_REWRITE 플래그 없이 복제해 보십시오(잘못 복제).
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("파일이 복제 되었습니다!");
      else
         Print("파일이 복제되지 않았습니다!");
      //--- InpDst 파일의 내용이 그대로 남아있습니다
      FileDisplay(InpDst);
      //--- FILE_REWRITE 플래그를 사용하여 다시 한 번 복제합니다 (파일이 있는 경우 올바른 복제
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("파일이 복제 되었습니다!");
      else
         Print("파일이 복제되지 않았습니다!");
     }
//--- InpSrc 파일 복제본 수령
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------+
//| 파일 내용 읽기                                                   |
//+------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- 에러값 리셋
   ResetLastError();
//--- 파일 열기
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- 파일 내용을 루프에 표시
      Print("+----------------------+");
      PrintFormat("File name = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      Print("+----------------------+");
      //--- 파일 닫기
```

© 2000-2025, MetaQuotes Ltd.

```
      FileClose(file_handle);
      return(true);
     }
//--- 파일을 열지 못했습니다
   PrintFormat("%s is not opened, error = %d",file_name,GetLastError());
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileDelete

클라이언트 터미널의 로컬 폴더에 있는 지정된 파일을 삭제합니다.

```
bool  FileDelete(
   const string  file_name,     // 삭제할 파일명
   int           common_flag=0  // 삭제할 파일의 위치
   );
```

Parameters

file_name

[in] 파일명.

common_flag=0

[in] 파일 위치를 결정짓는 플래그. common_flag = FILE_COMMON 라면, 파일은 모든 클라이언트 터미널의 공유 폴더인 ₩Terminal₩Common₩Files에 있습니다. 그렇지 않으면, 파일은 로컬 폴더에 있습니다.

반환 값

오류가 발생하면 함수는 false를 반환합니다.

참고

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

클라이언트 터미널의 로컬 폴더에 있는 지정된 파일을 삭제합니다 (MQL5₩Files 또는 테스트의 경우 MQL5₩Tester₩Files). common_flag = FILE_COMMON, 라면, 파일은 모든 클라이언트 터미널의 공유 폴더인 \Terminal\Common\Files에 있습니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 오래된 파일의 날짜
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // 파일 이름 저장 변수
   string   filter="*.txt"; // 파일 검색을 위한 필터
   datetime create_date;    // 파일 생성일
   string   files[];        // 파일명 목록
   int      def_size=25;    // 기본 배열 규모
   int      size=0;         // 파일 수
//--- 배열용 메모리 할당
   ArrayResize(files,def_size);
//--- 로컬 폴더의 루트에서 검색 핸들을 수신
   long search_handle=FileFindFirst(filter,file_name);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- FileFindFirst()이 성공적으로 실행한 경우 검증
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 루프테어 파일 검색
      do
        {
         files[size]=file_name;
         //--- 배열 규모를 늘림
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- 에러값 리셋
         ResetLastError();
         //--- 파일 생성일 수령
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- 파일이 오래되었는지 검증
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s 파일 삭제됨!",file_name);
            //--- 오래된 파일 삭제
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- 검색 핸들 닫기
      FileFindClose(search_handle);
     }
   else
     {
      Print("파일을 찾을 수 없습니다!");
      return;
     }
//--- 남아 있는 파일 확인
   PrintFormat("결과:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s 파일 존재!",files[i]);
      else
         PrintFormat("%s 파일 삭제됨!",files[i]);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

FileMove

로컬 또는 공유 폴더에서 다른 폴더로 파일을 이동합니다.

```
bool  FileMove(
   const string  src_file_name,     // 이동 작업의 파일 이름
   int           common_flag,       // 위치
   const string  dst_file_name,     // 대상 파일의 이름
   int           mode_flags         // 엑세스 모드
   );
```

Parameters

src_file_name

[in] 이동/이름변경할 파일명.

common_flag

[in] 파일 위치를 결정짓는 플래그. common_flag = FILE_COMMON 라면, 파일은 모든 클라이언트 터미널의 공유 폴더인 ₩Terminal₩Common₩Files에 있습니다. 그렇지 않으면 파일이 로컬 폴더에 있습니다(*common_flag=0*).

dst_file_name

[in] 작업 후의 파일명

mode_flags

[in] 엑세스 플래그. 매개 변수에는 FILE_REWRITE 및/또는 FILE_COMMON 2개의 플래그만 포함될 수 있습니다. 다른 플래그는 무시됩니다. 파일이 이미 존재하고 FILE_REWRITE 플래그를 지정하지 않으면 파일이 다시 작성되지 않고 함수가 false를 반환합니다.

반환 값

오류가 발생하면 함수는 false를 반환합니다.

참고

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

새 파일이 이미 있는 경우 mode_flags 매개 변수에서 FILE_REWRITE 플래그가 사용 가능한지 여부에 따라 복제본이 만들어집니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 입력 매개 변수
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- 파일 경로 수령
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- 원본 파일이 있는지 확인 (그렇지않으면 - exit)
   if(FileIsExist(src_path))
      PrintFormat("%s 파일이 %s\\Files\\%s 폴더에 있습니다",InpSrcName,local,InpSrcDirect
   else
     {
      PrintFormat("에러, %s 소스 파일이 탐색되지 않습니다",InpSrcName);
      return;
     }
//--- 결과 파일이 이미 있는지 확인
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("%s 파일이 %s\\Files\\%s 폴더에 존재합니다",InpDstName,common,InpDstDir
      //--- 파일이 존재하고, 이동은 FILE_REWRITE 플래그로 수행되어야 합니다.
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("%s 파일이 이동됐습니다",InpSrcName);
      else
         PrintFormat("에러! Code = %d",GetLastError());
     }
   else
     {
      PrintFormat("%s 파일이 %s\\Files\\%s 폴더에 존재하지 않습니다",InpDstName,common,Inp
      //--- 파일이 없으므로 FILE_REWRITE 플래그 없이 이동해야 합니다
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("%s 파일이 이동됐습니다",InpSrcName);
      else
         PrintFormat("에러! Code = %d",GetLastError());
     }
//--- 파일이 이동되었습니다; 확인해 보겠습니다.
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("성공!");
   else
      Print("오류!");
  }
```

더 보기

FileIsExist

© 2000-2025, MetaQuotes Ltd.

FileFlush

입력/출력 파일 버퍼에 남아 있는 모든 데이터를 디스크에 씁니다.

```
void  FileFlush(
   int  file_handle      // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

값이 반환되지 않음.

참고

파일에 쓸 때 데이터는 실제로 시간이 좀 지난 후에만 찾을 수 있습니다. 파일에 데이터를 즉시 저장하려면 FileFlush() 기능을 사용하십시오. 이 기능을 사용하지 않으면 아직 디스크에 저장되지 않은 데이터의 일부가 FileClose() 기능을 사용하여 파일을 닫을 때만 강제로 기록됩니다.

이 기능은 기록된 데이터가 특정 값일 때 사용해야 합니다. 자주 기능 호출이 프로그램 작동 속도에 영향을 줄 수 있다는 점을 유념해야 합니다.

파일에서 파일을 읽고 파일에 쓰는 작업 사이에 FileFlush() 기능을 호출해야 합니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 쓰기를 위한 파일명
input string InpFileName="example.csv"; // 파일명
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 에러값 리셋
   ResetLastError();
//--- 파일 열기
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- 파일에 데이터 쓰기
      for(int i=0;i<1000;i++)
        {
         //--- 쓰기 함수 호출
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sy
         //--- 128번째 반복 시마다 디스크에 데이터를 저장합니다.
         if((i & 127)==127)
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            //--- 이제 데이터는 파일에 저장되며 심각한 오류가 발생해도 손실되지 않습니다.
            FileFlush(file_handle);
            PrintFormat("i = %d, OK",i);
           }
         //--- 0.01 초 일시정지
         Sleep(10);
        }
      //--- 파일 닫기
      FileClose(file_handle);
     }
   else
      PrintFormat("Error, code = %d",GetLastError());
  }
```

더 보기

FileClose

© 2000-2025, MetaQuotes Ltd.

FileGetInteger

파일의 정수 속성을 가져옵니다. 이 함수에는 두 가지 변수가 있습니다.

1. 파일 핸들로 속성을 가져옵니다.

```
long  FileGetInteger(
   int                          file_handle,    // 파일 핸들
   ENUM_FILE_PROPERTY_INTEGER   property_id     // 속성 ID
   );
```

2. 파일 이름으로 속성을 가져옵니다.

```
long  FileGetInteger(
   const string                 file_name,            // 파일명
   ENUM_FILE_PROPERTY_INTEGER   property_id,          // 속성 ID
   bool                         common_folder=false   // 파일이 로컬 폴더에 표시됨 (false)
   );                                                 // 아니면 모든 터미널의 공통 폴더에 (
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

file_name

[in] 파일명.

property_id

[in] 파일 속성 ID. 이 값은 ENUM_FILE_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다. 함수의 두번째 변수가 사용되는 경우 다음 속성의 값만 수신할 수 있습니다: FILE_EXISTS, FILE_CREATE_DATE, FILE_MODIFY_DATE, FILE_ACCESS_DATE and FILE_SIZE.

common_folder=false

[in] 파일 위치를 가리킴. 매개 변수가 false이면 터미널 데이터 폴더가 표시됩니다. 그렇지 않으면 파일이 모든 터미널의 공유폴더인 ₩Terminal₩Common₩Files (FILE_COMMON)에 있다고 가정합니다.

반환 값

속성 값. 오류가 발생하면 -1이 반환됩니다. 오류 코드를 가져오려면 GetLastError() 함수를 사용합니다.

이름으로 속성을 가져올 때 폴더를 지정하면 반환 값은 정확하지만 함수에 오류 5018(ERR_MQL_FILE_IS_DIRECTORY)이 발생합니다.

참고

이 함수는 항상 오류 코드를 변경합니다. 성공적으로 완료되면 오류 코드가 NULL로 재설정됩니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 입력 매개 변수
input string InpFileName="data.csv";
```

© 2000-2025, MetaQuotes Ltd.

```
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- 파일 열기
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- 파일에 관한 모든 정보 출력
      Print(InpFileName," 파일 정보:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"date");
      FileInfo(handle,FILE_MODIFY_DATE,l,"date");
      FileInfo(handle,FILE_ACCESS_DATE,l,"date");
      FileInfo(handle,FILE_SIZE,l,"other");
      FileInfo(handle,FILE_POSITION,l,"other");
      FileInfo(handle,FILE_END,l,"bool");
      FileInfo(handle,FILE_IS_COMMON,l,"bool");
      FileInfo(handle,FILE_IS_TEXT,l,"bool");
      FileInfo(handle,FILE_IS_BINARY,l,"bool");
      FileInfo(handle,FILE_IS_CSV,l,"bool");
      FileInfo(handle,FILE_IS_ANSI,l,"bool");
      FileInfo(handle,FILE_IS_READABLE,l,"bool");
      FileInfo(handle,FILE_IS_WRITABLE,l,"bool");
      //--- 파일 닫기
      FileClose(handle);
     }
   else
      PrintFormat("%s file is not opened, ErrorCode = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------+
//| 파일 속성 값 표시                                                |
//+------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- 속성값 수령
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
      //--- 수신된 값을 올바른 형식으로 표시
      if(!StringCompare(type,"bool"))
         Print(EnumToString(id)," = ",l ? "true" : "false");
      if(!StringCompare(type,"date"))
```

© 2000-2025, MetaQuotes Ltd.

```
         Print(EnumToString(id)," = ",(datetime)l);
      if(!StringCompare(type,"other"))
         Print(EnumToString(id)," = ",l);
     }
   else
      Print("Error, Code = ",GetLastError());
  }
```

더 보기

File Operations, File Properties

© 2000-2025, MetaQuotes Ltd.

FileIsEnding

읽는 과정에서 파일의 끝을 정의합니다.

```
bool  FileIsEnding(
   int  file_handle      // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

이 함수는 파일 포인터를 읽거나 이동하는 과정에서 파일 끝에 도달하면 true를 반환합니다.

참고

파일의 끝을 정의하기 위해 함수는 파일의 다음 문자열 읽기를 시도합니다. 문자열이 없으면 함수는 true를 반환하고, 그렇지 않으면 false를 반환합니다.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 입력 매개 변수
input string InpFileName="file.txt";     // 파일명
input string InpDirectoryName="Data";    // 디렉토리명
input int    InpEncodingType=FILE_ANSI; // ANSI =32 또는 UNICODE=64
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 사용할 파일의 경로를 인쇄
   PrintFormat("Working %s\\Files\\ folder",TerminalInfoString(TERMINAL_DATA_PATH));
//--- 에러값 리셋
   ResetLastError();
//--- 읽기 목적으로 파일 열기(파일이 없는 경우 오류 발생).
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      //--- 파일 내용 인쇄
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- 파일 닫기
      FileClose(file_handle);
     }
   else
      PrintFormat("Error, code = %d",GetLastError());
  }
```

© 2000-2025, MetaQuotes Ltd.

FileIsLineEnding

읽는 과정에서 텍스트 파일의 줄 끝을 정의합니다.

```
bool  FileIsLineEnding(
   int  file_handle      // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

txt 또는 csv-file을 읽는 과정에서 줄 끝에 도달하면 true를 반환합니다(문자 CR-LF).

예 (FileWriteString 함수에 대한 예제를 실행하는 동안 얻은 파일이 여기에 사용됨)

```
//+------------------------------------------------------------------+
//|                                        Demo_FileIsLineEnding.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- 데이터 읽기를 위한 파라미터
input string InpFileName="RSI.csv";   // 파일명
input string InpDirectoryName="Data"; // 디렉토리명
//--- 지표 버퍼
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- 과매수 변수
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- 과매도 변수
int      ovs_ind=0;
```

© 2000-2025, MetaQuotes Ltd.

```
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 기본적 배열 크기 변수
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- 배열용 메모리 할당
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_DATA_PATH));
      double value;
      //--- 파일에서 데이터 읽기
      while(!FileIsEnding(file_handle))
        {
         //--- 문자열의 첫 값 읽기
         value=FileReadNumber(file_handle);
         //--- 함수 결과에 따라 다른 배열로 읽음
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
     {
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 배열 바인딩
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- 차트에 표시되지 않는 지표 값을 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 파일의 문자열 데이터 읽기                                        |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- 문자열 또는 파일의 끝에 도달할 때까지 읽기
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- 수를 보고 포지션 변경
      if(flag)
         FileReadNumber(file_handle);
      //--- 현재 날짜 저장
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- 필요시 배열 규모 증가
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- 첫 번째 반복을 지나침
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- 아직 다루지 않은 막대의 루프
   for(int i=prev_calculated;i<rates_total;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 디폴트로 0
       open_buff[i]=0;
       high_buff[i]=0;
       low_buff[i]=0;
       close_buff[i]=0;
       color_buff[i]=0;
       //--- 데이터가 아직 있는지 확인
       if(ovb_ind<ovb_size)
          for(int j=ovb_ind;j<ovb_size;j++)
            {
             //--- 날짜가 일치하면 막대는 과매수 영역에 있음
             if(time[i]==ovb_time[j])
               {
                open_buff[i]=open[i];
                high_buff[i]=high[i];
                low_buff[i]=low[i];
                close_buff[i]=close[i];
                //--- 0 - 적색
                color_buff[i]=0;
                //--- 카운터 증가
                ovb_ind=j+1;
                break;
               }
            }
       //--- 데이터가 여전히 존재하는지 확인
       if(ovs_ind<ovs_size)
          for(int j=ovs_ind;j<ovs_size;j++)
            {
             //--- 날짜가 일치하면 막대는 과매도된 영역에 있음
             if(time[i]==ovs_time[j])
               {
                open_buff[i]=open[i];
                high_buff[i]=high[i];
                low_buff[i]=low[i];
                close_buff[i]=close[i];
                //--- 1 - 청색
                color_buff[i]=1;
                //--- 카운터 증가
                ovs_ind=j+1;
                break;
               }
            }
      }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| ChartEvent 이벤트 핸들러                                         |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//--- 틱에 따라 지표의 폭을 변경
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

더 보기

FileWriteString

© 2000-2025, MetaQuotes Ltd.

FileReadArray

문자열을 제외한 모든 유형의 BIN 유형 배열 파일에서 읽습니다(문자열 포함 안 함, 동적 배열일 수 있음).).

```
uint  FileReadArray(
   int     file_handle,              // 파일 핸들
   void&   array[],                  // 기록할 배열
   int     start=0,                  // 쓸 배열 포지션 시작
   int     count=WHOLE_ARRAY         // 읽기 카운트
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

array[]

[out] 데이터를 로드할 배열.

start=0

[in] 배열에서 읽을 시작 위치.

count=WHOLE_ARRAY

[in] 읽을 요소 수. 기본적으로 전체 배열을 읽습니다. (count=WHOLE_ARRAY).

반환 값

읽은 요소 수.

참고

문자열 배열은 TXT 유형의 파일에서만 읽을 수 있습니다. 필요한 경우 함수는 어레이의 크기를 늘립니다.

예 (FileWriteArray 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 입력 매개 변수
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| 가격 데이터 저장 구조                                            |
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // 날짜
   double            bid;  // 입찰가
   double            ask;  // 요청가
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 구조 배열
   prices arr[];
//--- 파일 경로
   string path=InpDirectoryName+"//"+InpFileName;
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- 파일에서 배열로 모든 데이터를 읽음
      FileReadArray(file_handle,arr);
      //--- 배열 규모 수령
      int size=ArraySize(arr);
      //--- 배열로부터 데이터 출력
      for(int i=0;i<size;i++)
         Print("Date = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
      Print("Total data = ",size);
      //--- 파일 닫기
      FileClose(file_handle);
     }
   else
      Print("파일 열기 실패, 오류 ",GetLastError());
  }
```

더 보기

Variables, FileWriteArray

© 2000-2025, MetaQuotes Ltd.

FileReadBool

CSV 유형 문자열 파일을 현재 위치에서 구분 기호(또는 텍스트 줄 끝까지)로 읽고 읽기 문자열을 부울 유형 값으로 변환합니다.

```
bool  FileReadBool(
   int  file_handle    // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

줄 읽기는 "true", "false" 또는 정수 "0" 또는 "1"의 심볼 표시로 설정할 수 있습니다. 0이 아닌 값은 논리적 참으로 변환됩니다. 함수는 변환된 값을 반환합니다.

예 (FileWrite 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//+------------------------------------------------------------------+
//|                                              Demo_FileReadBool.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- 데이터 읽기를 위한 파라미터
input string InpFileName="MACD.csv";  // 파일명
input string InpDirectoryName="Data"; // 디렉토리명
//--- 글로벌 변수
int      ind=0;       // 인덱스
double   upbuff[];    // 위쪽 화살표의 지표 버퍼
double   downbuff[];  // 아래쪽 화살표의 지표 버퍼
```

© 2000-2025, MetaQuotes Ltd.

```
bool     sign_buff[]; // 시그널 배열 (true - 매수, false - 매도)
datetime time_buff[]; // 시그널 도착 시간 배열
int      size=0;      // 시그널 배열의 크기
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일이 읽기용으로 열렸습니다",InpFileName);
      //--- 첫째, 신호 수를 읽기
      size=(int)FileReadNumber(file_handle);
      //--- 배열용 메모리 할당
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- 파일에서 데이터 읽기
      for(int i=0;i<size;i++)
        {
         //--- 시그널 시간
         time_buff[i]=FileReadDatetime(file_handle);
         //--- 시그널 값
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- 파일 닫기
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 배열 바인딩
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- PLOT_ARROW에 그리기 위한 심볼 코드를 설정합니다.
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- 차트에 표시되지 않는 지표 값을 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(high,false);
//--- 아직 다루지 않은 막대의 루프
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 디폴트로 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- 데이터가 아직 있는지 확인
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 날짜가 일치하는 경우 파일의 값을 사용
            if(time[i]==time_buff[j])
              {
               //--- 시그널에 따라 화살표를 그림
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- 카운터 증가
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

더 보기

Type bool, FileWrite

© 2000-2025, MetaQuotes Ltd.

FileReadDatetime

CSV 파일에서 다음 형식 중 하나의 문자열 읽기: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" 또는 "HH:MI:SS" - 그리고 datetime 유형의 값으로 변환.

```
datetime  FileReadDatetime(
   int  file_handle    // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

datetime 유형 값.

예 (FileWrite 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//+------------------------------------------------------------------+
//|                                      Demo_FileReadDateTime.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- 데이터 읽기를 위한 파라미터
input string InpFileName="MACD.csv";  // 파일명
input string InpDirectoryName="Data"; // 디렉토리명
//--- 글로벌 변수
int      ind=0;       // 인덱스
double   upbuff[];    // 위쪽 화살표의 지표 버퍼
double   downbuff[];  // 아래쪽 화살표의 지표 버퍼
bool     sign_buff[]; // 시그널 배열 (true - 매수, false - 매도)
```

© 2000-2025, MetaQuotes Ltd.

```
datetime time_buff[]; // 시그널 도착 시간 배열
int      size=0;      // 시그널 배열의 크기
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일이 읽기용으로 열렸습니다",InpFileName);
      //--- 첫째, 신호 수를 읽기
      size=(int)FileReadNumber(file_handle);
      //--- 배열용 메모리 할당
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- 파일에서 데이터 읽기
      for(int i=0;i<size;i++)
        {
         //--- 시그널 시간
         time_buff[i]=FileReadDatetime(file_handle);
         //--- 시그널 값
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- 파일 닫기
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 배열 바인딩
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- PLOT_ARROW에 그리기 위한 심볼 코드를 설정합니다.
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- 차트에 표시되지 않는 지표 값을 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(high,false);
//--- 아직 다루지 않은 막대의 루프
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 디폴트로 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- 데이터가 아직 있는지 확인
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 날짜가 일치하는 경우 파일의 값을 사용
            if(time[i]==time_buff[j])
              {
               //--- 시그널에 따라 화살표를 그림
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- 카운터 증가
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

더 보기

Type datetime, StringToTime, TimeToString, FileWrite

© 2000-2025, MetaQuotes Ltd.

FileReadDouble

이진 파일의 현재 위치에서 이중 정밀 부동 소수점 번호(double)를 읽습니다.

```
double  FileReadDouble(
   int  file_handle    // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

double 타입의 값.

참고

오류에 대한 더 많은 정보에 대해선, GetLastError()을 호출하십시오.

예 (FileWriteDouble 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- 데이터 읽기 매개 변수
input string InpFileName="MA.csv";     // file name
input string InpDirectoryName="Data"; // 디렉토리명
//--- 글로벌 변수
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- 지표 버퍼
double   buff[];
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 커스텀 지표 초기화 기능                                  |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 먼저, 파일의 데이터 양을 읽습니다
      size=(int)FileReadDouble(file_handle);
      //--- 배열용 메모리 할당
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- 파일에서 데이터 읽기
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
     {
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 인덱스가 0인 지표 버퍼에 배열 바인딩
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- 차트에 표시되지 않는 지표 값을 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                    |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- 아직 다루지 않은 막대의 루프
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 디폴트로 0
      buff[i]=0;
      //--- 데이터가 여전히 존재하는지 확인
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 날짜가 일치하면 파일의 값이 사용됩니다
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //--- 카운터 증가
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

더 보기

Real types (double, float), StringToDouble, DoubleToString, FileWriteDouble

© 2000-2025, MetaQuotes Ltd.

FileReadFloat

이진 파일의 현재 위치에서 단일 정밀 부동 소수점 번호(float)를 읽습니다.

```
float  FileReadFloat(
   int  file_handle    // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

float 타입의 값.

참고

오류에 대한 더 많은 정보에 대해선, GetLastError()을 호출하십시오.

예 (FileWriteFloat 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//+------------------------------------------------------------------+
//|                                            Demo_FileReadFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- 데이터 읽기를 위한 파라미터
input string InpFileName="Close.bin"; // 파일명
input string InpDirectoryName="Data"; // 디렉토리명
//--- 글로벌 변수
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- 지표 버퍼
double   buff[];
double   color_buff[];
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- 어레이에 메모리 할당
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 파일에서 데이터 읽기
      while(!FileIsEnding(file_handle))
        {
         //--- 시간과 가격에 대한 값 읽기
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- 오버플로인 경우 어레이 크기를 늘립니다
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터를 읽습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
     {
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 배열을 지표 버퍼에 바인딩
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- 차트에 표시되지 않는 지표 값을 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- 아직 다루지 않은 막대의 루프
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 디폴트로 0
      buff[i]=0;
      color_buff[i]=0; // 디폴트로 적색
      //--- 데이터가 아직 있는지 확인
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 날짜가 일치하면 파일의 값이 사용됩니다
            if(time[i]==time_buff[j])
              {
               //--- 가격 수령
               buff[i]=close_buff[j];
               //--- 현재 가격이 이전 가격을 초과할 경우 색상은 파란색이 됩니다
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- 카운터 증가
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

더 보기

Real types (double, float), FileReadDouble, FileWriteFloat

© 2000-2025, MetaQuotes Ltd.

FileReadInteger

이 함수는 파일 포인터의 현재 위치에서 바이트 단위로 지정된 길이에 따라 int, short 또는 char 값을 읽습니다.

```
int  FileReadInteger(
   int  file_handle,         // 파일 핸들
   int  size=INT_VALUE       // 정수의 바이트 단위의 사이즈
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

size=INT_VALUE

[in] 읽어야 하는 바이트 수 (최대 4). 해당하는 상수가 제공됩니다: CHAR_VALUE = 1, SHORT_VALUE = 2 및 INT_VALUE = 4, 따라서 함수는 char, short 또는 int 유형의 전체 값을 읽을 수 있습니다.

반환 값

int 유형의 값 이 기능의 결과는 대상 유형, 즉 읽어야 하는 데이터 유형에 명시적으로 캐스트해야 합니다. int 유형의 값이 반환되므로 모든 정수 값으로 쉽게 변환할 수 있습니다. 파일 포인터가 읽은 바이트 수에 따라 이동됩니다.

참고

읽기가 4바이트 미만일 경우 수신된 결과는 항상 양수입니다. 1 또는 2 바이트를 읽을 경우 숫자 기호는 문자(1바이트) 또는 짧은 문자(2바이트)로 명시적으로 캐스팅하여 확인할 수 있습니다. 해당하는 underlying 타입이 없으므로 3바이트 번호에 대한 기호를 가져오는 것이 간단하지 않습니다..

예 (FileWriteInteger 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//+------------------------------------------------------------------+
//|                                        Demo_FileReadInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 데이터 읽기를 위한 파라미터
input string InpFileName="Trend.bin"; // file name
```

© 2000-2025, MetaQuotes Ltd.

```
input string InpDirectoryName="Data"; // 디렉토리명
//--- 글로벌 변수
int      ind=0;
int      size=0;
datetime time_buff[];
//--- 지표 버퍼
double   buff[];
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- 배열용 메모리 할당
   ArrayResize(time_buff,def_size);
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 부가 변수
      int    arr_size;
      uchar  arr[];
      //--- 파일에서 데이터 읽기
      while(!FileIsEnding(file_handle))
        {
         //--- 시간을 작성하는데 얼마나 많은 심볼이 사용되는지 확인
         arr_size=FileReadInteger(file_handle,INT_VALUE);
         ArrayResize(arr,arr_size);
         for(int i=0;i<arr_size;i++)
            arr[i]=(char)FileReadInteger(file_handle,CHAR_VALUE);
         //--- 시간 값 저장
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- 어레이가 오버플로되면 어레이의 크기 증가
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터를 읽습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 배열을 지표 버퍼에 바인딩
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- 차트에 표시되지 않는 지표 값을 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(close,false);
//--- 아직 다루지 않은 막대의 루프
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 디폴트로 0
      buff[i]=0;
      //--- 데이터가 아직 있는지 확인
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 날짜가 일치하면 파일의 값이 사용됩니다
            if(time[i]==time_buff[j])
              {
               //--- 가격 수령
               buff[i]=close[i];
               //--- 카운터 증가
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
```

© 2000-2025, MetaQuotes Ltd.

```
   return(rates_total);
  }
```

더 보기

IntegerToString, StringToInteger, Integer types, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileReadLong

이 함수는 이진 파일의 현재 위치에서 long 유형의 정수(8바이트)를 읽습니다.

```
long  FileReadLong(
   int  file_handle    // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

long 타입의 값.

예 (FileWriteLong 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//+------------------------------------------------------------------+
//|                                            Demo_FileReadLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Volume"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrYellow
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- 데이터 읽기를 위한 파라미터
input string InpFileName="Volume.bin"; // 파일명
input string InpDirectoryName="Data";  // 디렉토리명
//--- 글로벌 변수
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- 지표 버퍼
double   buff[];
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일이 쓰기를 위해 열렸습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 먼저, 파일의 데이터 양을 읽습니다
      size=(int)FileReadLong(file_handle);
      //--- 배열용 메모리 할당
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- 파일에서 데이터 읽기
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터를 읽습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
     {
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 0 인덱스를 사용하여 배열을 지표 버퍼에 바인딩
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- 차트에 표시될 지표 값을 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 아직 다루지 않은 막대의 루프
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 디폴트로 0
      buff[i]=0;
      //--- 데이터가 아직 있는지 확인
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 날짜가 일치하면 파일의 값이 사용됩니다
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

더 보기

Integer types, FileReadInteger, FileWriteLong

© 2000-2025, MetaQuotes Ltd.

FileReadNumber

이 함수는 CSV 파일에서 현재 위치에서 구분 기호(또는 텍스트 문자열 끝까지)까지 문자열을 읽고 읽기 문자열을 double 유형의 값으로 변환합니다.

```
double  FileReadNumber(
   int  file_handle    // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

double 타입의 값.

예 (FileWriteString 함수에 대한 예제를 실행하는 동안 얻은 파일이 여기에 사용됨)

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadNumber.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- 데이터 읽기를 위한 파라미터
input string InpFileName="RSI.csv";   // 파일명
input string InpDirectoryName="Data"; // 디렉토리명
//--- 지표 버퍼
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- 과매수 변수
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- 과매도 변수
```

© 2000-2025, MetaQuotes Ltd.

```
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 기본적 배열 크기 변수
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- 배열용 메모리 할당
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_DATA_PATH));
      double value;
      //--- 파일에서 데이터 읽기
      while(!FileIsEnding(file_handle))
        {
         //--- 문자열의 첫 값 읽기
         value=FileReadNumber(file_handle);
         //--- 함수 결과에 따라 다른 배열로 읽음
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
     {
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- 배열 바인딩
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
   SetIndexBuffer(4,color_buff,INDICATOR_COLOR_INDEX);
//---- 차트에 표시되지 않는 지표 값을 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 파일의 문자열 데이터 읽기                                        |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- 문자열 또는 파일의 끝에 도달할 때까지 읽기
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- 번호를 읽은 후 carriage 시프트
      if(flag)
         FileReadNumber(file_handle);
      //--- 현재 날짜 저장
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- 필요시 배열 규모 증가
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- 첫 번째 반복을 지나침
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- 아직 다루지 않은 막대의 루프
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 디폴트로 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- 기한이 남아 있는지 확인
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- 날짜가 일치하면 막대는 과매수 영역에 있음
            if(time[i]==ovb_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 0 - 적색
               color_buff[i]=0;
               //--- 카운터 증가
               ovb_ind=j+1;
               break;
              }
           }
      //--- 데이터가 여전히 존재하는지 확인
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- 날짜가 일치하면 막대는 과매도된 영역에 있음
            if(time[i]==ovs_time[j])
              {
               open_buff[i]=open[i];
               high_buff[i]=high[i];
               low_buff[i]=low[i];
               close_buff[i]=close[i];
               //--- 1 - 청색
               color_buff[i]=1;
               //--- 카운터 증가
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| ChartEvent 이벤트 핸들러                                         |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//--- 틱에 따라 지표의 폭을 변경
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

더 보기

FileWriteString

© 2000-2025, MetaQuotes Ltd.

FileReadString

이 함수는 파일에 있는 파일 포인터의 현재 위치에서 문자열을 읽습니다.

```
string  FileReadString(
   int  file_handle,     // 파일 핸들
   int  length=-1        // 문자열 길이
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

length=-1

[in] 읽을 문자 수.

반환 값

Line read (string).

참고

bin 파일에서 읽을 때. 읽을 문자열의 길이를 지정해야 합니다. txt 파일에서 읽을 때는 문자열 길이가 필요하지 않으며, 문자열은 현재 위치에서 라인 피드 문자 "₩₩"까지 읽힙니다. CSV 파일에서 읽을 때는 문자열 길이도 필요하지 않습니다. 문자열은 현재 위치에서 가장 가까운 구분 기호 또는 텍스트 문자열 끝 문자까지 읽힙니다.

파일이FILE_ANSI 플래그로 열리면, 라인 읽기가 Unicode로 변환됩니다.

예 (FileWriteInteger 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 데이터 읽기를 위한 파라미터
input string InpFileName="Trend.bin"; // file name
input string InpDirectoryName="Data"; // 디렉토리명
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 부가 변수
      int    str_size;
      string str;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 파일에서 데이터 읽기
      while(!FileIsEnding(file_handle))
        {
         //--- 시간을 작성하는데 얼마나 많은 심볼이 사용되는지 확인
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- 문자열 읽기
         str=FileReadString(file_handle,str_size);
         //--- 문자열 출력
         PrintFormat(str);
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터를 읽습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
  }
```

더 보기

String Type, Conversion Functions, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileReadStruct

이 함수는 파일 포인터의 현재 위치부터 시작하여 이진 파일에서 파라미터로 전달된 구조로 내용을 읽습니다.

```
uint  FileReadStruct(
   int           file_handle,        // 파일 핸들
   const void&   struct_object,      // 내용을 읽을 대상 구조
   int           size=-1             // 구조 크기(바이트)
   );
```

Parameters

file_handle

[in] 열린 bin 파일의 파일 설명자.

struct_object

[out] 이 구조의 개체. 구조에는 문자열, 동적 배열 또는 가상 함수가 포함되어서는 안 됩니다.

size=-1

[in] 읽어져야 할 바이트 수. 크기가 지정되지 않았거나 표시된 값이 구조의 크기보다 큰 경우 지정된 구조의 정확한 크기가 사용됩니다.

반환 값

성공하면 함수는 읽은 바이트 수를 반환합니다. 파일 포인터가 동일한 바이트 수만큼 이동합니다.

예 (FileWriteStruct 함수에 대한 예제를 실행한 후 얻은 파일이 여기에 사용됩니다)

```
//+------------------------------------------------------------------+
//|                                          Demo_FileReadStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Candles"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- 데이터 수신에 대한 매개 변수
input string  InpFileName="EURUSD.txt"; // 파일명
input string  InpDirectoryName="Data";  // 디렉토리명
//+------------------------------------------------------------------+
//| 캔들스틱 데이터 저장 구조                                        |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // 시가
   double            close; // 종가
   double            high;  // 고가
   double            low;   // 저가
   datetime          date;  // 날짜
  };
//--- 지표 버퍼
double       open_buff[];
double       close_buff[];
double       high_buff[];
double       low_buff[];
//--- 글로벌 변수
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 기능                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- 파일 열기
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_COMMONDATA_PATH)
      //--- 파일에서 데이터 읽기
      while(!FileIsEnding(file_handle))
        {
         //--- 배열에 데이터 작성
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
         //--- 배열이 오버플로 되었는지 확인
         if(size==default_size)
           {
            //--- 배열 크기 증가
            default_size+=100;
            ArrayResize(cand_buff,default_size);
           }
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터를 읽습니다, %s 파일이 닫힙니다",InpFileName);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
    else
      {
       PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
       return(INIT_FAILED);
      }
//--- 지표 버퍼 연결
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- 빈(empty) 값
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   ArraySetAsSeries(time,false);
//--- 아직 처리되지 않은 캔들스틱의 루프
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- 디폴트로 0
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //--- 데이터가 아직 있는지 확인
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- 날짜가 일치하면 파일의 값이 사용됩니다
            if(time[i]==cand_buff[j].date)
              {
               open_buff[i]=cand_buff[j].open;
               close_buff[i]=cand_buff[j].close;
```

© 2000-2025, MetaQuotes Ltd.

```
                high_buff[i]=cand_buff[j].high;
                low_buff[i]=cand_buff[j].low;
                //--- 카운터 증가
                ind=j+1;
                break;
               }
            }
         }
      }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

더 보기

Structures and classes, FileWriteStruct

© 2000-2025, MetaQuotes Ltd.

FileSeek

이 함수는 지정된 포지션을 기준으로 파일 포인터의 포지션을 지정된 바이트 수만큼 이동합니다.

```
bool  FileSeek(
   int                  file_handle,     // 파일 핸들
   long                 offset,          // 바이트 단위
   ENUM_FILE_POSITION   origin           // 레퍼런스용 포지션
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

offset

[in] 바이트 이동(음수 값이 될 수 있음).

origin

[in] 변위의 시작점. ENUM_FILE_POSITION 열거값 중 하나일 수 있습니다.

반환 값

성공하면 함수는 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 더 많은 정보에 대해선 GetLastError() 함수를 호출하십시오.

참고

FileSeek() 함수를 실행하면 파일의 "레벨 경계"를 벗어나는 음의 이동이 발생하는 경우 파일 포인터가 파일 시작 부분으로 설정됩니다.

위치가 파일의 "오른쪽 경계"를 벗어나 설정된 경우(파일 크기보다 큼), 다음 번 파일 쓰기는 파일 끝이 아니라 포지션 세트에서 수행됩니다. 이 경우 이전 파일 끝과 포지션 세트에 대해 무한값이 기록됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSeek.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 입력 매개 변수
input string InpFileName="file.txt";     // 파일명
input string InpDirectoryName="Data";    // 디렉토리명
input int    InpEncodingType=FILE_ANSI; // ANSI =32 또는 UNICODE=64
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 난수 생성을 위한 변수 값 지정
   _RandomSeed=GetTickCount();
//--- 문자열의 시작점 위치에 대한 변수
   ulong pos[];
   int   size;
//--- 에러값 리셋
   ResetLastError();
//--- 파일 열기
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
      //--- 파일의 각 문자열에 대한 시작 위치 수신
      GetStringPositions(file_handle,pos);
      //--- 파일의 문자열 수 정의
      size=ArraySize(pos);
      if(!size)
        {
         //--- 파일에 문자열이 없으면 중지
         PrintFormat("%s 파일이 비었습니다!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- 문자열 수를 무작위로 고름
      int ind=MathRand()%size;
      //--- 위치를 문자열의 시작점으로 이동
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- ind 숫자로 문자열을 읽고 인쇄합니다.
         PrintFormat("%d 번호 포함 문자열 텍스트: ₩%s\"",ind,FileReadString(file_handle
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("%s 파일이 닫혔습니다",InpFileName);
     }
   else
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------------------+
//| 함수는 파일의 각 문자열에 대한 시작 지점을 정의하고  |
//| 배열에 배치합니다                                                            |
//+------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- 디폴트 배열 사이즈
   int def_size=127;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 배열용 메모리 할당
   ArrayResize(arr,def_size);
//--- 스트링 카운터
   int i=0;
//--- 파일의 끝이 아니라면 문자열이 하나 이상 있습니다
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // 파일이 비었습니다, exit
//--- 인코딩에 따른 바이트 이동 정의
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- 루프에서 문자열을 통과
   while(1)
     {
      //--- string 읽기
      FileReadString(handle);
      //--- 파일의 끝 점검
      if(!FileIsEnding(handle))
        {
         //--- 다음 문자열의 위치를 저장
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- 오버플로인 경우 어레이의 크기를 증가
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // 파일의 끝, exit
     }
//--- 배열의 실제 사이즈 정의
   ArrayResize(arr,i);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileSize

이 함수는 파일 크기(바이트)를 반환합니다.

```
ulong  FileSize(
   int  file_handle    // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

int 타입의 값.

참고

오류에 대한 더 많은 정보에 대해선, GetLastError()을 호출하십시오.

예:

```
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 입력 매개 변수
input ulong  InpThresholdSize=20;        // file threshold size in kilobytes
input string InpBigFolderName="big";     // 큰 파일을 위한 폴더
input string InpSmallFolderName="small"; // 작은 파일을 위한 폴더
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // 파일 이름 저장 변수
   string   filter="*.csv"; // 파일 검색을 위한 필터
   ulong    file_size=0;    // 바이트 단위의 파일 사이즈
   int      size=0;         // 파일 수
//--- 사용할 파일의 경로를 인쇄
   PrintFormat("Working in %s\\Files\\ folder",TerminalInfoString(TERMINAL_COMMONDATA_
//--- 모든 터미널의 공통 폴더 루트에 있는 검색 핸들을 수신합니다
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- FileFindFirst()가 성공적으로 실행되었는지 확인
   if(search_handle!=INVALID_HANDLE)
     {
      //--- 크기에 따라 루프의 파일을 이동
      do
        {
         //--- 파일 오픈
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
         if(file_handle!=INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             //--- 파일 크기 수신
             file_size=FileSize(file_handle);
             //--- 파일 닫기
             FileClose(file_handle);
            }
          else
            {
             PrintFormat("%s 파일 오픈 실패, 에러 코드 = %d",file_name,GetLastError());
             continue;
            }
          //--- 파일 사이즈 출력
          PrintFormat("%s 파일 사이즈가 %d 바이트와 같습니다",file_name,file_size);
          //--- 파일 이동 경로 정의
          string path;
          if(file_size>InpThresholdSize*1024)
             path=InpBigFolderName+"//"+file_name;
          else
             path=InpSmallFolderName+"//"+file_name;
          //--- 파일 이동
          ResetLastError();
          if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
             PrintFormat("%s 파일이 이동되었습니다",file_name);
          else
             PrintFormat("Error, code = %d",GetLastError());
         }
       while(FileFindNext(search_handle,file_name));
       //--- 검색 핸들 닫기
       FileFindClose(search_handle);
      }
   else
       Print("파일을 찾을 수 없습니다!");
  }
```

© 2000-2025, MetaQuotes Ltd.

FileTell

이 파일은 열린 파일의 현재 파일 포인터 위치를 반환합니다.

```
ulong  FileTell(
   int  file_handle    // 파일 핸들
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

반환 값

파일 설명자의 현재 위치(바이트 단위).

참고

오류에 대한 더 많은 정보에 대해선, GetLastError()을 호출하십시오.

예:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileTell.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 입력 매개 변수
input string InpFileName="file.txt";     // 파일명
input string InpDirectoryName="Data";    // 디렉토리명
input int    InpEncodingType=FILE_ANSI; // ANSI =32 또는 UNICODE=64
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 난수 생성을 위한 변수 값 지정
   _RandomSeed=GetTickCount();
//--- 문자열의 시작점 위치에 대한 변수
   ulong pos[];
   int   size;
//--- 에러값 리셋
   ResetLastError();
//--- 파일 열기
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("%s 파일을 읽을 수 있습니다",InpFileName);
       //--- 파일의 각 문자열에 대한 시작 위치 수신
       GetStringPositions(file_handle,pos);
       //--- 파일의 문자열 수 정의
       size=ArraySize(pos);
       if(!size)
         {
          //--- 파일에 문자열이 없으면 중지
          PrintFormat("%s 파일이 비었습니다!",InpFileName);
          FileClose(file_handle);
          return;
         }
       //--- 문자열 수를 무작위로 고름
       int ind=MathRand()%size;
       //--- 위치를 문자열의 시작점으로 이동
       FileSeek(file_handle,pos[ind],SEEK_SET);
       //--- ind 숫자로 문자열을 읽고 인쇄합니다.
       PrintFormat("String text with %d number: \"%s\"",ind,FileReadString(file_handle));
       //--- 파일 닫기
       FileClose(file_handle);
       PrintFormat("%s 파일이 닫혔습니다",InpFileName);
      }
    else
       PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------------------+
//| 함수는 파일의 각 문자열에 대한 시작 지점을 정의하고  |
//| 배열에 배치합니다                                                              |
//+------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- 디폴트 배열 사이즈
   int def_size=127;
//--- 배열용 메모리 할당
   ArrayResize(arr,def_size);
//--- 스트링 카운터
   int i=0;
//--- 파일의 끝이 아니라면 문자열이 하나 이상 있습니다
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // 파일이 비었습니다, exit
//--- 인코딩에 따른 바이트 이동 정의
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
```

© 2000-2025, MetaQuotes Ltd.

```
      shift=1;
   else
      shift=2;
//--- 루프에서 문자열을 통과
   while(1)
     {
      //--- string 읽기
      FileReadString(handle);
      //--- 파일의 끝 점검
      if(!FileIsEnding(handle))
        {
         //--- 다음 문자열의 위치를 저장
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- 오버플로인 경우 어레이의 크기를 증가
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // 파일의 끝, exit
     }
//--- 배열의 실제 사이즈 정의
   ArrayResize(arr,i);
  }
```

© 2000-2025, MetaQuotes Ltd.

FileWrite

이 함수는 데이터를 CSV 파일에 쓰기 위한 것으로, 0이 아닌 한 구분 기호가 자동으로 삽입됩니다. 파일에 쓴 후 줄 끝 문자 "₩₩"이 추가됩니다.

```
uint  FileWrite(
   int  file_handle,   // 파일 핸들
   ...                 // 기록된 파라미터 목록
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

...

[in] 파일에 쓸 매개 변수 목록(콤마로 구분). 쓰여진 매개 변수 수는 최대 63개까지 가능합니다.

반환 값

쓴 바이트 수.

참고

숫자는 출력 시 텍스트로 변환됩니다(Print() 함수 참조). double 유형의 데이터는 소수점 다음 16자리 정확도로 출력되며, 데이터는 가장 컴팩트한 형식에 따라 기존 형식 또는 과학적 형식으로 표시될 수 있습니다. float 유형의 데이터는 소수점 뒤에 5자리 숫자로 표시됩니다. 다른 정밀도 또는 명확하게 지정된 형식으로 실수를 출력하려면 DoubleToString()를 사용하십시오.

bool 유형의 번호는 "true" 또는 "false" 문자열로 표시됩니다. datetime 타입의 숫자는 "YYYY.MM.DD HH:MI:SS"로 표시됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileWrite.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 터미널에서 데이터를 수신하기 위한 매개 변수
input string             InpSymbolName="EURUSD";           // 통화
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // 타임프레임
input int                InpFastEMAPeriod=12;              // 빠른 EMA 기간
input int                InpSlowEMAPeriod=26;              // 느린 EMA 기간
input int                InpSignalPeriod=9;                // 차이 평균 기간
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // 가격 유형
input datetime           InpDateStart=D'2012.01.01 00:00'; // 데이터 복사 시작 날짜
//--- 파일에 데이터를 쓰기 위한 매개 변수
```

© 2000-2025, MetaQuotes Ltd.

```
input string             InpFileName="MACD.csv";  // 파일명
input string             InpDirectoryName="Data"; // 디렉토리명
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // 데이터 복사 종료일
   bool     sign_buff[]; // 시그널 배열 (true - 매수, false - 매도)
   datetime time_buff[]; // 시그널 배열의 도착 시간
   int      sign_size=0; // 시그널 배열 사이즈
   double   macd_buff[]; // 지표 값 배열
   datetime date_buff[]; // 지표 날짜 배열
   int      macd_size=0; // 지표 배열 사이즈
//--- 종료 시간은 현재 시간
   date_finish=TimeCurrent();
//--- MACD 지표 핸들 수신
   ResetLastError();
   int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeri
   if(macd_handle==INVALID_HANDLE)
     {
      //--- 지표 핸들을 수신하지 못함
      PrintFormat("지표 핸들 수신 시 오류 발생. Error code = %d",GetLastError());
      return;
     }
//--- 지표가 모든 값을 계산할 때까지 루프에 있음
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // 지표가 모든 값을 계산할 수 있도록 일시 중지
//--- 일정 기간 동안 지표 값을 복제
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("지표값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 지표값에 적절한 시간을 복제
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("시간 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 지표의 메모리를 비움
   IndicatorRelease(macd_handle);
//--- 버퍼 사이즈 수령
   macd_size=ArraySize(macd_buff);
//--- 데이터를 분석하고 지표 시그널을 배열에 저장합니다
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=1;i<macd_size;i++)
     {
      //--- 매수 시그널
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- 매도 시그널
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- 지표 값을 쓰기 위해 파일을 엽니다(파일이 없을 경우 자동으로 생성됨)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 쓸 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 첫째, 시그널의 수를 씁니다
      FileWrite(file_handle,sign_size);
      //--- 시그널에 대한 시간 및 값을 파일에 기록
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
  }
```

더 보기

Comment, Print, StringFormat

© 2000-2025, MetaQuotes Ltd.

FileWriteArray

이 함수는 문자열을 제외한 모든 유형의 어레이를 BIN 파일에 씁니다(스트링 또는 동적 어레이가 포함되지 않은 구조의 어레이일 수 있음).

```
uint  FileWriteArray(
   int           file_handle,          // 파일 핸들
   const void&   array[],              // 배열
   int           start=0,              // 배열의 시작 인덱스
   int           count=WHOLE_ARRAY     // 요소 수
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

array[]

[out] 기록용 배열.

start=0

[in] 어레이의 초기 인덱스(첫 번째로 기록된 요소의 수).

count=WHOLE_ARRAY

[in] 쓸 항목 수 (WHOLE_ARRAY는 숫자로 시작하는 모든 항목이 배열이 끝날 때까지 기록됨을 의미합니다).

반환 값

기록된 항목 수.

참고

문자열 배열을 TXT 파일에 기록할 수 있습니다. 이 경우 문자열은 줄 끝 문자 "₩r₩n"에 의해 자동으로 끝납니다. 파일 형식 ANSI 또는 UNICODE에 따라 문자열이 ANSI 인코딩으로 변환되거나 변환되지 않습니다.

예:

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteArray.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 입력 매개 변수
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| 가격 데이터 저장 구조                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
struct prices
  {
   datetime          date; // 날짜
   double            bid;  // 입찰가
   double            ask;  // 요청가
  };
//--- 글로벌 변수
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 배열용 메모리 할당
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 카운트가 n보다 작을 경우 남은 카운트를 씁니다
   WriteData(count);
  }
//+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- 배열에 데이터 저장
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- 현재 데이터를 보여줌
   Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask = ",arr[count].ask);
//--- 카운터 증가
   count++;
//--- 배열이 가득 찬 경우 파일에 데이터를 기록한 후 0으로 설정합니다
   if(count==size)
     {
      WriteData(size);
      count=0;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 파일에 배열의 요소 기록                                          |
//+------------------------------------------------------------------+
void WriteData(const int n)
  {
//--- 파일 열기
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
   if(handle!=INVALID_HANDLE)
     {
      //--- 파일 끝에 어레이 데이터를 씁니다
      FileSeek(handle,0,SEEK_END);
      FileWriteArray(handle,arr,0,n);
      //--- 파일 닫기
      FileClose(handle);
     }
   else
      Print("파일 오픈 실패, 오류 ",GetLastError());
  }
```

더 보기

Variables, FileSeek

© 2000-2025, MetaQuotes Ltd.

FileWriteDouble

이 함수는 파일 포인터의 현재 위치에서 시작하여 이중 파라미터 값을 빈 파일에 기록합니다.

```
uint  FileWriteDouble(
   int     file_handle,     // 파일 핸들
   double  value            // 작성할 값
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

value

[in] double 타입의 값.

반환 값

성공하면 함수는 쓴 바이트 수를 반환합니다 (이 경우 sizeof(double)=8). 파일 포인터가 동일한 바이트 수만큼 이동됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteDouble.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 터미널에서 데이터를 수신하기 위한 매개 변수
input string             InpSymbolName="EURJPY";           // 통화
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_M15;       // 타임프레임
input int                InpMAPeriod=10;                   // smoothing period
input int                InpMAShift=0;                     // 지표 시프트
input ENUM_MA_METHOD     InpMAMethod=MODE_SMA;             // smoothing type
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // 가격 유형
input datetime           InpDateStart=D'2013.01.01 00:00'; // 데이터 복사 시작 날짜
//--- 파일에 데이터를 쓰기 위한 매개 변수
input string             InpFileName="MA.csv";     // 파일명
input string             InpDirectoryName="Data"; // 디렉토리명
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- MA 지표 핸들 수신
   ResetLastError();
   int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,
   if(ma_handle==INVALID_HANDLE)
     {
      //--- 지표 핸들을 수신하지 못했습니다
      PrintFormat("지표 핸들 수신 시 오류 발생. Error code = %d",GetLastError());
      return;
     }
//--- 지표가 모든 값을 계산할 때까지 루프에 있음
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // 지표가 모든 값을 계산할 수 있도록 잠시 정지
   PrintFormat("%s에서 시작하는 지표 값이 파일에 기록됩니다",TimeToString(InpDateStart));
//--- 지표 값 복제
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("지표 값을 복사하지 못했습니다. Error code = %d",GetLastError());
      return;
     }
//--- 적절한 막대 도착 시간을 복제
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("시간 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 버퍼 사이즈 수령
   size=ArraySize(ma_buff);
//--- 지표의 메모리를 비움
   IndicatorRelease(ma_handle);
//--- 지표 값을 쓰기 위해 파일을 엽니다(파일이 없을 경우 자동으로 생성됨)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 쓸 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 첫째, 데이터 샘플의 크기를 씁니다
      FileWriteDouble(file_handle,(double)size);
      //--- 파일에 시간과 값을 기록
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
  }
```

더 보기

Real types (double, float)

© 2000-2025, MetaQuotes Ltd.

FileWriteFloat

이 함수는 파일 포인터의 현재 위치에서 시작하여 float 파라미터의 값을 빈 파일에 기록합니다.

```
uint  FileWriteFloat(
   int    file_handle,     // 파일 핸들
   float  value            // 작성될 값
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

value

[in] float 타입의 값.

반환 값

성공하면 함수는 쓴 바이트 수를 반환합니다 (이 경우 sizeof(float)=4). 파일 포인터가 동일한 바이트 수만큼 이동됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWriteFloat.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 터미널에서 데이터를 수신하기 위한 매개 변수
input string          InpSymbolName="EURUSD";           // 통화
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;       // 타임프레임
input datetime        InpDateStart=D'2013.01.01 00:00'; // 데이터 복사 시작 날짜
//--- 파일에 데이터를 쓰기 위한 매개 변수
input string          InpFileName="Close.bin"; // 파일명
input string          InpDirectoryName="Data"; // 디렉토리명
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- 에러값 리셋
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- 매 막대의 종가 복제
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("종가 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 매 막대의 시간 복제
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("시간 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 버퍼 사이즈 수령
   size=ArraySize(close_buff);
//--- 값 쓰기를 위해 파일 오픈(파일이 없을 경우 자동으로 생성됨)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일이 쓰기를 위해 열렸습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 종가 시간과 값을 파일에 기록
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
  }
```

더 보기

Real types (double, float), FileWriteDouble

© 2000-2025, MetaQuotes Ltd.

FileWriteInteger

함수는 파일 포인터의 현재 위치에서 시작하여 int 파라미터 값을 빈 파일에 기록합니다.

```
uint  FileWriteInteger(
   int  file_handle,         // 파일 핸들
   int  value,               // 작성될 값
   int  size=INT_VALUE       // 바이트 단위 사이즈
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

value

[in] 정수 값.

size=INT_VALUE

[in] 기록해야 하는 바이트 수(최대 4). 해당 상수가 제공됩니다: CHAR_VALUE=1, SHORT_VALUE=2 및 INT_VALUE=4, 따라서 함수는 char, uchar, short, usshort, int 또는 uint 유형의 정수 값을 쓸 수 있습니다.

반환 값

성공하면 함수는 쓴 바이트 수를 반환합니다. 파일 포인터가 동일한 바이트 수만큼 이동됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                      Demo_FileWriteInteger.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 터미널에서 데이터를 수신하기 위한 매개 변수
input string              InpSymbolName="EURUSD";           // 통화
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_H1;        // 타임프레임
input datetime            InpDateStart=D'2013.01.01 00:00'; // 데이터 복사 시작 날짜
//--- 파일에 데이터를 쓰기 위한 매개 변수
input string              InpFileName="Trend.bin"; // 파일명
input string              InpDirectoryName="Data"; // 디렉토리명
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
```

© 2000-2025, MetaQuotes Ltd.

```
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- 에러값 리셋
   ResetLastError();
//--- 매 막대의 종가 복제
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("종가의 값을 복사하지 못했습니다. Error code = %d",GetLastError());
      return;
     }
//--- 매 막대의 시간 복제
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("시간 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 버퍼 사이즈 수령
   size=ArraySize(close_buff);
//--- 값 쓰기를 위해 파일 오픈(파일이 없을 경우 자동으로 생성됨)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 쓸 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_DATA_PATH));
      //---
      int   up_down=0; // 추세 플래그
      int   arr_size;  // arr 배열 사이즈
      uchar arr[];     // uchar type array
      //--- 파일에 시간 값 작성
      for(int i=0;i<size-1;i++)
        {
         //--- 현재 막대와 다음 막대의 종가를 비교
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
               //--- FileWriteInteger를 사용하여 파일에 날짜 값 작성
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- 먼저 배열의 심볼 수를 기록합니다
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- 심볼 작성
               for(int j=0;j<arr_size;j++)
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- 추세 플래그 변경
               up_down=1;
              }
```

© 2000-2025, MetaQuotes Ltd.

```
            }
          else
            {
             if(up_down!=-1)
               {
                //--- FileWriteInteger를 사용하여 파일에 날짜 값 작성
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //--- 먼저 배열의 심볼 수를 기록합니다
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- 심볼 작성
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- 추세 플래그 변경
                up_down=-1;
               }
            }
        }
     //--- 파일 닫기
     FileClose(file_handle);
     PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
    }
  else
     PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
 }
```

더 보기

IntegerToString, StringToInteger, Integer types

© 2000-2025, MetaQuotes Ltd.

FileWriteLong

이 함수는 파일 포인터의 현재 위치에서 시작하여 long 타입 파라미터의 값을 빈(bin) 파일에 기록합니다.

```
uint  FileWriteLong(
   int    file_handle,     // 파일 핸들
   long   value            // 작성될 값
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

value

[in] long 타입 값.

반환 값

성공하면 함수는 쓴 바이트 수를 반환합니다(이 경우 sizeof(long)=8). 파일 포인터가 동일한 바이트 수만큼 이동됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                            Demo_FileWriteLong.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 터미널에서 데이터를 수신하기 위한 매개 변수
input string          InpSymbolName="EURUSD";           // 통화
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // 타임프레임
input datetime        InpDateStart=D'2013.01.01 00:00'; // 데이터 복사 시작 날짜
//--- 파일에 데이터를 쓰기 위한 매개 변수
input string          InpFileName="Volume.bin"; // 파일명
input string          InpDirectoryName="Data";  // 디렉토리명
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
//--- 에러값 리셋
```

© 2000-2025, MetaQuotes Ltd.

```
   ResetLastError();
//--- 각 막대에 대해 틱 볼륨 복제
   if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_bu
     {
      PrintFormat("틱 볼륨의 값 복사 실패. Error code = %d",GetLastError());
      return;
     }
//--- 매 막대의 시간 복제
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("시간 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 버퍼 사이즈 수령
   size=ArraySize(volume_buff);
//--- 지표 값을 쓰기 위해 파일을 엽니다(파일이 없을 경우 자동으로 생성됨)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일을 쓸 수 있습니다",InpFileName);
      PrintFormat("파일 경로: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- 먼저 데이터 샘플 크기를 작성합니다.
      FileWriteLong(file_handle,(long)size);
      //--- 파일에 시간 및 볼륨 값 쓰기
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
  }
```

더 보기

Integer types, FileWriteInteger

© 2000-2025, MetaQuotes Ltd.

FileWriteString

이 함수는 파일 포인터의 현재 위치에서 시작하여 문자열 유형 파라미터의 값을 BIN, CSV 또는 TXT 파일에 기록합니다. CSV 또는 TXT 파일에 쓸 때: 문자열 '₩n' (LF)에 이전 문자 '₩r' (CR)가 없는 심볼이 있으면 '₩n' 전에 누락된 '₩r'가 추가됩니다.

```
uint  FileWriteString(
   int           file_handle,     // 파일 핸들
   const string  text_string,     // 작성할 문자열
   int           length=-1        // 심볼의 수
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

text_string

[in] 문자열.

length=-1

[in] 쓸 문자 수. 이 옵션은 문자열을 BIN 파일에 쓰는 데 필요합니다. 크기가 지정되지 않은 경우, 트레일러 0이 없는 전체 문자열이 기록됩니다. 문자열 길이보다 작은 크기를 지정하면 트레일러 0이 없는 문자열의 일부가 기록됩니다. 문자열 길이보다 큰 크기를 지정하면 문자열이 적절한 수의 0으로 채워집니다. CSV 및 TXT 유형의 파일의 경우 이 매개 변수는 무시되고 문자열이 완전히 기록됩니다.

반환 값

성공하면 함수는 쓴 바이트 수를 반환합니다. 파일 포인터가 동일한 바이트 수만큼 이동됩니다.

참고

FILE_UNICODE 플래그로 열린 파일에 쓸 때(또는 FILE_ANSI 플래그 없이) 쓴 바이트 수는 기록된 문자열 문자 수보다 두 배 더 큽니다. FILE_ANSI 플래그로 열린 파일에 기록할 때, 쓴 바이트 수는 기록된 문자열 문자의 수와 일치합니다.

예:

```
//+------------------------------------------------------------------+
//|                                        Demo_FileWriteString.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 터미널에서 데이터를 수신하기 위한 매개 변수
input string             InpSymbolName="EURUSD";           // 통화
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // 타임프레임
input int                InpMAPeriod=14;                   // MA period
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // 가격 유형
```

© 2000-2025, MetaQuotes Ltd.

```
input datetime           InpDateStart=D'2013.01.01 00:00'; // 데이터 복사 시작 날짜
//--- 파일에 데이터를 쓰기 위한 매개 변수
input string             InpFileName="RSI.csv";   // 파일명
input string             InpDirectoryName="Data"; // 디렉토리명
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // 데이터 복사 종료일
   double   rsi_buff[];  // 지표 값 배열
   datetime date_buff[]; // 지표 날짜 배열
   int      rsi_size=0;  // 지표 배열 사이즈
//--- 종료 시간은 현재 시간입니다
   date_finish=TimeCurrent();
//--- RSI 지표 핸들 수령
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- 지표 핸들을 수신하지 못했습니다
      PrintFormat("지표 핸들 수신 시 오류 발생. Error code = %d",GetLastError());
      return;
     }
//--- 지표가 모든 값을 계산할 때까지 루프에 있음
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // 지표가 모든 값을 계산할 수 있도록 잠시 정지
//--- 일정 기간 동안 지표 값을 복제
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("지표값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 지표값에 적절한 시간을 복제
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("시간 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 지표의 메모리를 비움
   IndicatorRelease(rsi_handle);
//--- 버퍼 사이즈 수령
   rsi_size=ArraySize(rsi_buff);
//--- 지표 값을 쓰기 위해 파일을 엽니다(파일이 없을 경우 자동으로 생성됨)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
```

© 2000-2025, MetaQuotes Ltd.

```
    {
     PrintFormat("%s 파일을 쓸 수 있습니다",InpFileName);
     PrintFormat("파일 경로: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
     //--- 변수를 추가로 준비
     string str="";
     bool   is_formed=false;
     //--- 과매수 및 과매도 영역의 형성 날짜 작성
     for(int i=0;i<rsi_size;i++)
       {
        //--- 지표 값 확인
        if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
          {
           //--- 값이 이 영역에서 첫 번째일 경우
           if(!is_formed)
             {
              //--- 값과 날짜를 더함
              str=(string)rsi_buff[i]+"\t"+(string)date_buff[i];
              is_formed=true;
             }
           else
              str+="\t"+(string)rsi_buff[i]+"\t"+(string)date_buff[i];
           //--- 다음 루프 반복으로 이동
           continue;
          }
        //--- 플래그 확인
        if(is_formed)
          {
           //--- 문자열이 형성되면, 파일에 작성
           FileWriteString(file_handle,str+"\r\n");
           is_formed=false;
          }
       }
     //--- 파일 닫기
     FileClose(file_handle);
     PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
    }
  else
     PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
}
```

더 보기

String Type, StringFormat

© 2000-2025, MetaQuotes Ltd.

FileWriteStruct

이 함수는 파일 포인터의 현재 위치에서 시작하여 파라미터로 전달된 구조의 빈(bin) 파일 내용에 씁니다.

```
uint  FileWriteStruct(
   int           file_handle,       // 파일 핸들
   const void&   struct_object,     // 개체에 연결
   int           size=-1            // 바이트 단위로 작성될 사이즈
   );
```

Parameters

file_handle

[in] FileOpen()에서 반환된 파일 설명자.

struct_object

[in] 이 구조의 객체에 대한 참조. 구조에는 문자열, 동적 배열 또는 가상 함수가 포함되어서는 안 됩니다.

size=-1

[in] 기록할 바이트 수. 크기가 지정되지 않았거나 지정된 바이트 수가 구조의 크기보다 크면 전체 구조가 기록됩니다.

반환 값

성공하면 함수는 쓴 바이트 수를 반환합니다. 파일 포인터가 동일한 바이트 수만큼 이동됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                          Demo_FileWiteStruct.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 스크립트를 시작할 때 입력 매개 변수의 창을 표시합니다.
#property script_show_inputs
//--- 터미널에서 데이터를 수신하기 위한 매개 변수
input string          InpSymbolName="EURUSD";           // 통화
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // 타임프레임
input datetime        InpDateStart=D'2013.01.01 00:00'; // 데이터 복사 시작 날짜
//--- 파일에 데이터를 쓰기 위한 매개 변수
input string          InpFileName="EURUSD.txt";         // 파일명
input string          InpDirectoryName="Data";          // 디렉토리명
//+------------------------------------------------------------------+
//| 캔들스틱 데이터 저장 구조                                        |
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // 시가
```

© 2000-2025, MetaQuotes Ltd.

```
   double            close; // 종가
   double            high;  // 고가
   double            low;   // 저가
   datetime          date;  // 날짜
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime     date_finish=TimeCurrent();
   int          size;
   datetime     time_buff[];
   double       open_buff[];
   double       close_buff[];
   double       high_buff[];
   double       low_buff[];
   candlesticks cand_buff[];
//--- 에러값 리셋
   ResetLastError();
//--- 범위에서 막대의 도착 시간을 수령
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("시간 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 범위에서 막대의 고가를 수령
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
     {
      PrintFormat("고가의 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 범위로부터 막대의 저가를 수령
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("저가의 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 범위로부터 막대 시가 수령
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
     {
      PrintFormat("시가 값 복제 실패. Error code = %d",GetLastError());
      return;
     }
//--- 범위로부터 막대의 종가 수령
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("종가의 값 복제 실패. Error code = %d",GetLastError());
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 배열 차원 정의
   size=ArraySize(time_buff);
//--- 구조 배열에 모든 데이터 저장
   ArrayResize(cand_buff,size);
   for(int i=0;i<size;i++)
     {
      cand_buff[i].open=open_buff[i];
      cand_buff[i].close=close_buff[i];
      cand_buff[i].high=high_buff[i];
      cand_buff[i].low=low_buff[i];
      cand_buff[i].date=time_buff[i];
     }

//--- 구조 어레이를 파일에 쓰기 위해 파일을 엽니다(파일이 없는 경우 자동으로 생성됨)
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s 파일이 쓰기를 위해 열렸습니다",InpFileName);
      PrintFormat("파일 경로: %s₩₩Files₩₩",TerminalInfoString(TERMINAL_COMMONDATA_PATH)
      //--- 바이트 수 카운터를 준비
      uint counter=0;
      //--- 루프에 어레이 값 쓰기
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat("%d 바이트의 정보가 %s 파일에 기록됩니다",InpFileName,counter);
      PrintFormat("총 바이트 수: %d * %d * %d = %d, %s",size,5,8,size*5*8,size*5*8==cou
      //--- 파일 닫기
      FileClose(file_handle);
      PrintFormat("데이터가 작성됐습니다, %s 파일이 닫힙니다",InpFileName);
     }
   else
      PrintFormat("%s 파일 열기 실패, 에러 코드 = %d",InpFileName,GetLastError());
  }
```

더 보기

Structures and classes

© 2000-2025, MetaQuotes Ltd.

FileLoad

지정된 이진 파일의 모든 데이터를 전달된 일련의 숫자 유형 또는 단순 구조로 읽습니다. 이 함수를 사용하여 알려진 유형의 데이터를 적절한 배열로 빠르게 읽을 수 있습니다.

```
long  FileLoad(
   const string  file_name,          // 파일명
   void&         buffer[],           // 숫자 유형 또는 단순 구조 배열
   int           common_flag=0       // 파일 플래그는 기본적으로 <data_folder>\MQL5\Files\
   );
```

Parameters

file_name

[in] 데이터를 읽을 파일의 이름.

buffer

[out] 숫자 유형 또는 단순 구조 배열.

common_flag=0

[in] 작업 모드를 나타내는 파일 플래그. 매개 변수를 지정하지 않으면 하위폴더 MQL5\Files (또는 테스트의 경우 <testing_agent_directory>\MQL5\Files)에서 파일이 검색됩니다.

반환 값

읽은 요소의 수 또는 오류 발생 시 -1.

참고

FileLoad() 함수는 파일에서 배열 요소 크기의 배수인 바이트 수를 읽습니다. 파일 크기가 10바이트이고 함수가 데이터를 double (sizeof(double)=8) 타입의 배열로 읽는다고 가정해봅시다. 이 경우 함수는 8바이트만 읽고 파일 끝에 있는 나머지 2바이트는 삭제되며 FileLoad() 함수는 1(1 요소 읽기)을 반환합니다.

예:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileLoad.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- 입력 매개 변수
input int      bars_to_save=10; // 막대의 수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_rates.bin";
   MqlRates rates[];
//---
   int copied=CopyRates(_Symbol,_Period,0,bars_to_save,rates);
   if(copied!=-1)
     {
      PrintFormat(" CopyRates(%s) copied %d bars",_Symbol,copied);
      //---  파일에 값 쓰기
      if(!FileSave(filename,rates,FILE_COMMON))
         PrintFormat("FileSave() 실패, error=%d",GetLastError());
     }
   else
      PrintFormat("Failed CopyRates(%s), error=",_Symbol,GetLastError());
//--- 이제 이 값을 다시 파일로 읽어보겠습니다.
   ArrayFree(rates);
   long count=FileLoad(filename,rates,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tOpen\tHigh\tLow\tClose\tTick Voulme\tSpread\tReal Volume");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s\t%G\t%G\t%G\t%G\t%I64u\t%d\t%I64u",
                     TimeToString(rates[i].time,TIME_DATE|TIME_SECONDS),
                     rates[i].open,rates[i].high,rates[i].low,rates[i].close,
                     rates[i].tick_volume,rates[i].spread,rates[i].real_volume);
        }
     }
  }
```

더 보기

Structures and Classes, FileReadArray, FileReadStruct, FileSave

© 2000-2025, MetaQuotes Ltd.

FileSave

매개 변수로 전달된 배열의 모든 요소를 이진 파일에 씁니다. 이 함수를 사용하여 숫자 타입 또는 단순 구조의 배열을 하나의 문자열로 빠르게 작성할 수 있습니다.

```
bool  FileSave(
   const string  file_name,          // 파일명
   void&         buffer[],           // 숫자 유형 또는 단순 구조 배열
   int           common_flag=0       // ㅏ파일 플래그(기본적으로 파일)은 <data_folder>\MQL5
   );
```

Parameters

file_name

[in] 데이터 배열에 파일명이 기록됩니다.

buffer

[in] 숫자 타입 또는 단순 구조의 배열.

common_flag=0

[in] 작업 모드를 나타내는 파일 플래그. 매개 변수를 지정하지 않으면 파일이 하위 폴더 MQL5\Files(또는 테스트의 경우 <testing_agent_directory>\MQL5\Files)에 기록됩니다.

반환 값

실패 시 False를 반환합니다.

예:

```
//+------------------------------------------------------------------+
//|                                                Demo_FileSave.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs
//--- 입력 매개 변수
input int      ticks_to_save=1000; // Number of ticks
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_ticks.bin";
   MqlTick ticks[];
u//---
   int copied=CopyTicks(_Symbol,ticks,COPY_TICKS_ALL,0,ticks_to_save);
   if(copied!=-1)
     {
      PrintFormat(" CopyTicks(%s)가 %d 틱을 복제했습니다",_Symbol,copied);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 틱 기록이 동기화되면 오류 코드가 0이 됩니다
      if(!GetLastError()==0)
         PrintFormat("%s: 틱이 동기화되지 않았습니다, error=%d",_Symbol,copied,_LastError
      //---  파일에 틱 쓰기
      if(!FileSave(filename,ticks,FILE_COMMON))
         PrintFormat("FileSave() 실패, error=%d",GetLastError());
     }
   else
      PrintFormat("CopyTicks(%s) 실패, Error=",_Symbol,GetLastError());
//--- 이제 틱을 다시 파일로 읽어봅니다
   ArrayFree(ticks);
   long count=FileLoad(filename,ticks,FILE_COMMON);
   if(count!=-1)
     {
      Print("Time\tBid\tAsk\tLast\tVolume\tms\tflags");
      for(int i=0;i<count;i++)
        {
         PrintFormat("%s.%03I64u:\t%G\t%G\t%G\t%I64u\t0x%04x",
         TimeToString(ticks[i].time,TIME_DATE|TIME_SECONDS),ticks[i].time_msc%1000,
         ticks[i].bid,ticks[i].ask,ticks[i].last,ticks[i].volume,ticks[i].flags);
        }
     }
  }
```

더 보기

Structures and Classes, FileWriteArray, FileWriteStruct, FileLoad, FileWrite

© 2000-2025, MetaQuotes Ltd.

FolderCreate

files 폴더에 디렉터리를 생성합니다(common_flag 값에 따라 다름)

```
bool  FolderCreate(
   string  folder_name,       // 생성된 폴더 이름의 행
   int     common_flag=0      // 액션 영역
   );
```

Parameters

folder_name

[in] 생성할 디렉토리의 이름. 폴더의 상대 경로를 포함합니다.

common_flag=0

[in] 디렉토리 위치를 정의하는 플래그. common_flag=FILE_COMMON 이면, 디렉토리는 모든 클라이언트의 공용폴더인 terminals₩₩terminal₩₩common₩₩files 에 있습니다. 그렇지 않으면, 디렉토리가 로컬 폴더에 있습니다 (MQL5₩₩files 또는 테스트의 경우 MQL5₩₩tester₩₩files).

반환 값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

보안을 위해 MQL5 언어로 파일 작업을 엄격하게 제어합니다. MQL5 언어로 파일 작업에 사용되는 파일은 파일 샌드박스 외부에 위치할 수 없습니다.

예:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 설명
#property description "스크립트에는 FolderCreate() 애플리케이션 샘플이 표시됩니다."
#property description "외부 파라미터는 폴더를 생성하기 위한 디렉터리를 정의합니다."
#property description "폴더 구조는 스크립트를 실행한 후 생성됩니다"

//--- 스크립트 실행 중 입력 매개 변수의 표시 창
#property script_show_inputs
//--- 입력 매개 변수는 스크립트가 작동하는 폴더를 정의합니다
input bool     common_folder=false; // 모든 터미널을 위한 공용 폴더
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- MQL5₩Files 에 생성될 폴더
   string root_folder="Folder_A";
   if(CreateFolder(root_folder,common_folder))
     {
      //--- 그 안에 Child_Folder_B1 sub-folder 생성
```

© 2000-2025, MetaQuotes Ltd.

```
      string folder_B1="Child_Folder_B1";
      string path=root_folder+"\\"+folder_B1;          // 구조를 고려하여 폴더 이름을 생성
      if(CreateFolder(path,common_folder))
        {
         //--- 이 폴더에 하위 디렉터리를 3개 더 생성
         string folder_C11="Child_Folder_C11";
         string child_path=root_folder+"\\"+folder_C11;// 구조를 고려하여 폴더 이름을 생성
         CreateFolder(child_path,common_folder);
         //--- 두 번째 하위 디렉터리
         string folder_C12="Child_Folder_C12";
         child_path=root_folder+"\\"+folder_C12;
         CreateFolder(child_path,common_folder);

         //--- third sub-directory
         string folder_C13="Child_Folder_C13";
         child_path=root_folder+"\\"+folder_C13;
         CreateFolder(child_path,common_folder);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| 폴더를 만들고 관련 메시지를 표시해 보십시오                      |
//+------------------------------------------------------------------+
bool CreateFolder(string folder_path,bool common_flag)
  {
   int flag=common_flag?FILE_COMMON:0;
   string working_folder;
//--- common_flag 파라미터에 따라 전체 경로 정의
   if(common_flag)
      working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
   else
      working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- 디버깅 메시지
   PrintFormat("folder_path=%s",folder_path);
//--- MQL5₩Files 경로와 관련한 폴더 생성 시도
   if(FolderCreate(folder_path,flag))
     {
      //--- 생성된 폴더의 전체 경로 표시
      PrintFormat("Created the folder %s",working_folder+"\\"+folder_path);
      //--- 에러 코드 재설정
      ResetLastError();
      //--- 실행 성공
      return true;
     }
   else
      PrintFormat("%s 폴더 생성 실패. Error code %d",working_folder+folder_path,GetLast
//--- 실행 실패
   return false;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

더 보기

FileOpen(), FolderClean(), FileCopy()

© 2000-2025, MetaQuotes Ltd.

FolderDelete

함수는 지정된 디렉토리를 제거합니다. 폴더가 비어 있지 않으면 제거할 수 없습니다.

```
bool  FolderDelete(
   string  folder_name,       // 삭제할 폴더 이름이 포함된 문자열
   int     common_flag=0      // 범위
   );
```

Parameters

folder_name

[in] 삭제할 디렉터리의 이름. 폴더의 전체 경로를 포함합니다.

common_flag=0

[in] 디렉토리 위치를 결정하는 플래그. common_flag=FILE_COMMON 라면, 디렉토리는 모든 클라이언트 터미널의 공유 폴더인 terminals₩₩terminal₩₩common₩₩files에 있습니다. 그렇지 않으면, 디렉토리는 로컬 폴더(MQL5₩₩files 또는 테스트의 경우 MQL5₩₩tester₩₩files)에 있습니다.

반환 값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

디렉토리에 하나 이상의 파일 및/또는 하위 디렉토리가 포함되어 있으면 이 디렉토리를 삭제할 수 없으므로 먼저 지워야 합니다. FolderClean()는 폴더의 모든 파일 또는 하위 폴더를 지우는 데 사용됩니다.

예:

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderDelete.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 설명
#property description "이 스크립트는 FolderDelete()의 샘플 사용을 보여 줍니다."
#property description "처음 두 개의 폴더가 생성되고, 그 중 하나는 비어 있고, 두 번째 폴더는 파
#property description "비어 있지 않은 폴더를 삭제하려고 하면 오류가 반환되고 경고가 표시됩니다.

//--- 스크립트를 시작할 때 입력 파라미터 대화 상자 표시
#property script_show_inputs
//--- 입력 파라미터
input string   firstFolder="empty";     // 빈(empty) 폴더
input string   secondFolder="nonempty";// 한 파일이 생성될 폴더
string filename="delete_me.txt";        // secondFolder 폴더에 생성할 파일명
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 여기에 파일 핸들을 쓰십시오
   int handle;
//--- 작업 중인 폴더 확인
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- 디버그 메시지
   PrintFormat("working_folder=%s",working_folder);
//--- 경로 MQL5\Files와 관련한 빈(empty) 폴더 만들기 시도 중
   if(FolderCreate(firstFolder,0)) // 0은 터미널의 로컬 폴더에서 작업 중임을 의미합니다
     {
      //--- 생성된 폴더의 전체 경로 입력
      PrintFormat("%s 폴더가 생성되었습니다",working_folder+"\\"+firstFolder);
      //--- 에러코드 재설정
      ResetLastError();
     }
   else
      PrintFormat("%s 폴더 생성 실패. Error code %d",working_folder+"\\"+firstFolder, G

//--- 이제 FileOpen() 함수를 사용하여 비어 있지 않은 폴더를 만드십시오
   string filepath=secondFolder+"\\"+filename;  // 존재하지 않는 폴더에 쓰기 위해 열려는 파
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // 이 경우 FILE_WRITE 플래그를 지정해야
   if(handle!=INVALID_HANDLE)
      PrintFormat("%s 파일이 읽기용으로 열렸습니다",working_folder+"\\"+filepath);
   else
      PrintFormat("%s 파일을 %s 폴더에 생성 실패. Error code=",filename,secondFolder, Get

   Comment(StringFormat("%s 와 %s 폴더 삭제 준비", firstFolder, secondFolder));
//--- 차트의 메시지를 읽기 위해 5초 정도 잠시 멈춤
   Sleep(5000); // Sleep()은 지표에서 사용되지 않습니다!

//--- 대화 상자를 표시하고 사용자에게 물음
   int choice=MessageBox(StringFormat("%s 및 %s 폴더를 삭제하시겠습니까?", firstFolder, s
                         "Deleting folders",
                         MB_YESNO|MB_ICONQUESTION); //  Two buttons - "Yes" and "No"

//--- 선택한 모델에 따라 작업 실행
   if(choice==IDYES)
     {
      //--- 차트에서 코멘트 삭제
      Comment("");
      //--- "Experts" 저널로 메시지 추가
      PrintFormat("%s 및 %s 폴더 삭제 시도",firstFolder, secondFolder);
      ResetLastError();
      //--- 빈(empty) 폴더 삭제
      if(FolderDelete(firstFolder))
         //--- 폴더가 비어 있으므로 다음 메시지가 표시되어야 합니다
         PrintFormat("%s 폴더가 성공적으로 삭제되었습니다",firstFolder);
```

© 2000-2025, MetaQuotes Ltd.

```
      else
         PrintFormat("%s 폴더 삭제 실패. Error code=%d", firstFolder, GetLastError());

      ResetLastError();
      //--- 파일이 들어 있는 폴더 삭제
      if(FolderDelete(secondFolder))
         PrintFormat("%s 폴더가 성공적으로 삭제됐습니다", secondFolder);
      else
         //--- 폴더에 파일이 포함되어 있으므로 다음 메시지가 표시되어야 합니다
         PrintFormat("%s 폴더 삭제 실패. Error code=%d", secondFolder, GetLastError());
     }
   else
      Print("삭제 취소");
//---
  }
```

더 보기

FileOpen(), FolderClean(), FileMove()

© 2000-2025, MetaQuotes Ltd.

FolderClean

이 함수는 지정된 폴더에 있는 모든 파일을 삭제합니다.

```
bool  FolderClean(
   string  folder_name,       // 삭제된 폴더명이 포함된 문자열
   int     common_flag=0      // 범위
   );
```

Parameters

folder_name

[in] 모든 파일을 삭제할 디렉터리의 이름. 폴더의 전체 경로를 포함합니다.

common_flag=0

[in] 디렉토리 위치를 결정하는 플래그. common_flag = FILE_COMMON 이면, 디렉토리는 모든 클라이언트 터미널의 공유 폴더인 ₩Terminal₩Common₩Files 에 있습니다. 그렇지 않으면 디렉터리가 로컬 폴더에 있습니다(MQL5₩Files 또는 테스트의 경우 MQL5₩Tester₩Files).

반환 값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

보안상의 이유로 파일 작업은 MQL5 언어로 엄격하게 제어됩니다. MQL5를 사용하여 파일 작업을 수행하는 파일은 파일 샌드박스 외부에 있을 수 없습니다.

이 함수는 모든 파일 및 모든 하위 디렉토리가 삭제될 수 있으므로 주의해야 합니다.

예:

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderClean.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 설명
#property description "이 스크립트는 FolderClean()의 샘플 사용을 보여 줍니다."
#property description "먼저 FileOpen() 함수를 사용하여 지정된 폴더에 파일을 생성합니다."
#property description "그런 다음 파일을 삭제하기 전에 MessageBox()를 사용하여 경고를 표시합니

//--- 스크립트를 시작할 때 입력 파라미터 대화 상자 표시
#property script_show_inputs
//--- 입력 파라미터
input string   foldername="demo_folder";  // MQL5/Files/ 에 폴더 생성
input int      files=5;                   // 생성 및 삭제할 파일 수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnStart()
  {
   string name="testfile";
//--- 터미널 데이터 폴더에서 먼저 파일 열기 또는 생성
   for(int N=0;N<files;N++)
     {
      //--- 'demo_folder\testfileN.txt'의 형태의 파일명
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- 쓰기 플래그가 있는 파일을 엽니다. 이 경우 'demo_folder'가 자동으로 생성됩니다.
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- FileOpen() 기능이 성공적이었는지 확인
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("%s 파일 생성 실패. Error code",filemane,GetLastError());
         ResetLastError();
        }
      else
        {
         PrintFormat("파일 %s이 성공적으로 열렸습니다",filemane);
         //--- 열려 있는 파일은 더 이상 필요하지 않으므로 닫으십시오
         FileClose(handle);
        }
     }

//--- 폴더의 파일 수 확인
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("폴더 %s에 모두 %d 파일이 포함되어 있습니다",foldername,k);

//--- 사용자에게 묻는 대화 상자 표시
   int choice=MessageBox(StringFormat("폴더 %s에서 파일 %d를 삭제하려고 하십니다. 계속 하시
                         "폴더에서 파일 삭제",
                         MB_YESNO|MB_ICONQUESTION); //  2개 버튼 - "예" 및 "아니오"
   ResetLastError();

//--- 선택한 변수모에 따라 작업 실행
   if(choice==IDYES)
     {
      //--- 파일 삭제 시작
      PrintFormat("폴더 %s에서 모든 파일 삭제 시도",foldername);
      if(FolderClean(foldername,0))
         PrintFormat("파일이 모두 삭제되어, %s 폴더에 %d 파일이 남았습니다",
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
         PrintFormat("폴더 %s에서 파일 삭제 실패. Error code %d",foldername,GetLastError
     }
   else
      PrintFormat("삭제 취소됨");
//---
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 지정된 폴더의 파일 수를 반환합니다                               |
//+------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- 하나 이상의 파일이 발견되면 추가 파일 검색
   if(handle!=INVALID_HANDLE)
     {
      //--- 파일명 보기
      PrintFormat("%s 파일 발견",filename);
      //--- 파일/폴더 발견 카운터 증가
      count++;
      //--- 모든 파일/폴더에서 탐색
      while(FileFindNext(handle,filename))
        {
         PrintFormat("%s 파일 발견",filename);
         count++;
        }
      //--- 완료 시 검색 핸들을 닫는 것을 잊지 마십시오
      FileFindClose(handle);
     }
   else // 핸들 가져오기 실패
     {
      PrintFormat("폴더 %s에서 파일 탐색 실패",path);
     }
//--- 결과 반환
   return count;
  }
```

더 보기

FileFindFirst, FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

고객 지표

사용자 지정 지표를 생성하는 데 사용되는 그룹 기능. 이러한 기능은 엑스퍼트 어드바이저 및 스크립트를 작성할 때 사용할 수 없습니다.

기능	액션
SetIndexBuffer	지정된 지표 버퍼를 다음 유형의 1차원 동적 배열 로 바인딩: double 유형
IndicatorSetDouble	다음 유형의 지표 속성 값을 설정: double 유형
IndicatorSetInteger	다음 유형의 지표 속성 값을 설정: int 유형
IndicatorSetString	다음 유형의 지표 속성 값을 설정: string 유형
PlotIndexSetDouble	다음 유형의 지시선 속성 값을 설정: double
PlotIndexSetInteger	다음 유형의 지시선 속성 값을 설정: int 유형
PlotIndexSetString	다음 유형의 지시선 속성 값을 설정: string 유형
PlotIndexGetInteger	다음 유형의 지시선 속성 값을 반환: 정수 유형

지표 속성은 컴파일러 지시 또는 함수를 사용하여 설정할 수 있습니다. 이를 보다 잘 이해하려면 예제의 지표 스타일을 학습하는 것이 좋습니다.

사용자 지정 지표의 모든 필요한 계산은 사전에 설정된 OnCalculate()함수에 배치되어야 합니다. OnCalculate() 기능 호출의 간단한 형식을 사용하는 다음과 같은 경우

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, co
```

rates_total 변수에는 지표 값 계산을 위한 입력 매개변수로 전달된 price[] 배열의 총 요소 수 값이 포함됩니다.

매개변수 *prev_calculated*는 이전 호출에서 OnCalculate()를 실행한 결과이며, 이를 통해 지표 값을 계산하기 위한 저장 알고리즘을 구성할 수 있습니다. 예를들어, 현재 값 rates_total = 1000, prev_calculated = 999, 이면 각 지표 버퍼의 값 하나만 계산하면 충분할 수 있습니다.

입력 배열 가격의 크기에 대한 정보를 사용할 수 없었다면 각 지표 버퍼의 1000개 값을 계산해야할 필요가 있었을 겁니다. OnCalculate() 값을 처음 호출할 때 prev_calculated = 0. price[] 배열이 변경된 경우 prev_calculated도 0입니다.

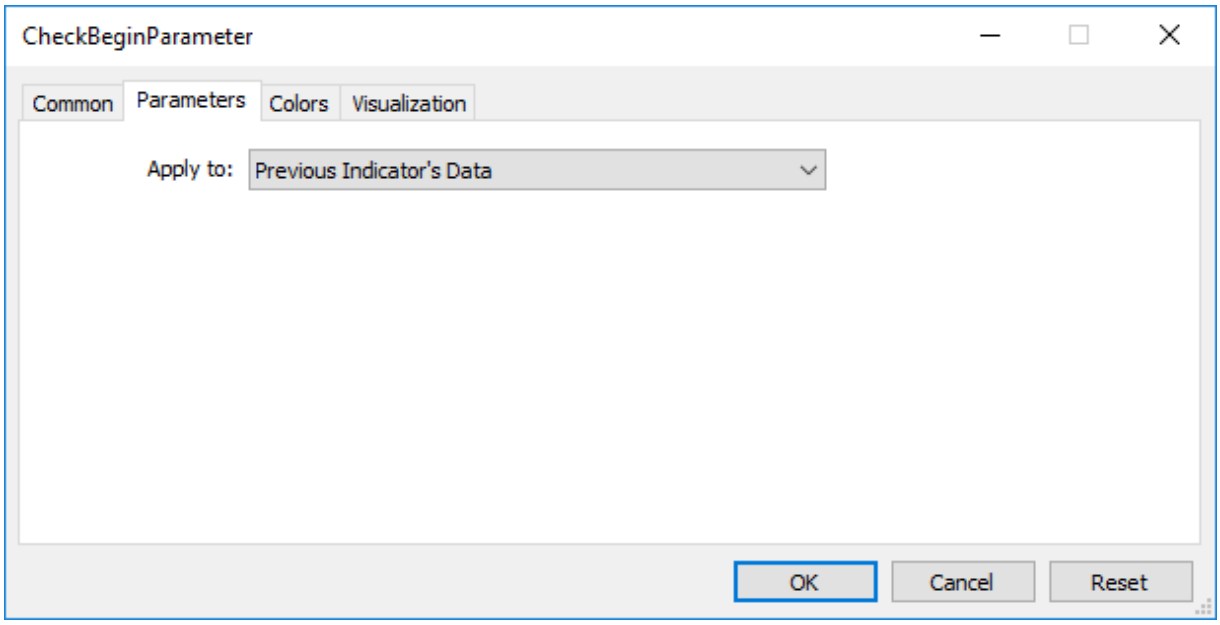
시작 매개변수는 계산을 위한 데이터가 없는 가격 배열의 초기 값 수를 나타냅니다. 예를 들어, 가속기 오실레이터 값(처음 37개 값이 계산되지 않음)을 입력 매개변수로 사용한 경우 = 37을 시작합니다. 예를 들어 다음과 같은 간단한 지표를 살펴보겠습니다:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_width1  1
//--- 지표 버퍼
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])

  {
//---
   Print("begin = ",begin,"  prev_calculated = ",prev_calculated,"  rates_total = ",ra
//--- 다음 번 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
```

"탐색기" 창에서 가속기 오실레이터 지표 창으로 끌어다 놓으면 이전 지표의 값을 기준으로 계산이 수행됨을 알 수 있습니다.



결과적으로, OnCalculate()의 첫 번째 호출은 prev_calculated의 값이 0이 되며, 추가 호출에서는 (가격 차트의 막대 수가 증가할 때까지) rates_total 값과 같을 것입니다.

© 2000-2025, MetaQuotes Ltd.

시작 매개변수의 값은 이 지표의 논리에 따라 가속기 지표의 값이 계산되지 않는 초기 막대 수와 정확히 같습니다. 사용자 지정 지표 Accelerator.mq5의 소스코드를 살펴보면 OnInit() 기능에 다음의 줄이 표시됩니다:

```
//--- 인덱스를 끌어올 첫 번째 막대를 설정
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values) 함수를 사용하여 사용자 지정 지표의 0 지표 배열에 존재하지 않는 첫 번째 값의 수를 설정했으며, 이 값은 계산에 사용할 필요가 없습니다 (empty_first_values). 따라서 다음과 같은 메커니즘이 있습니다:

1. 지표의 초기값 수를 설정하며, 이 값은 다른 사용자 지정 지표의 계산에 사용해서는 안 됩니다;
2. 계산 논리에 들어가지 않고 다른 사용자 지정 지표를 호출할 때 무시할 첫 번째 값의 수에 대한 정보를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

예제의 지표 스타일.

MetaTrader 5 클라이언트 터미널에는 적절한 기능을 사용하여 MQL5 프로그램에서 사용할 수 있는 38개의 기술 지표가 포함되어 있습니다. 그러나 MQL5 언어의 주요 장점은 사용자 지정 지표를 생성할 수 있다는 것이며, 이 지표를 엑스퍼트 어드바이저에서 사용하거나 기술 분석을 목적으로 가격 차트에 적용할 수 있습니다.

전체 지표 세트는 플롯이라고 하는 몇 가지 기본 그리기 스타일에서 파생될 수 있습니다. 플로팅은 지표가 요청에 따라 계산, 저장 및 제공하는 데이터를 표시하는 방법을 나타냅니다. 다음과 같은 7가지 기본 플롯 유형이 있습니다:

1. 선
2. 섹션(세그먼트)
3. 히스토그램
4. 화살표(심볼)
5. 그려진 영역(채워진 채널)
6. 막대
7. 일본 캔들스틱

각 플로팅에는 지표 값이 저장되는 double유형의 배열이 1~5개씩 필요합니다. 편의를 위해 이러한 배열은 지표 버퍼와 연결됩니다. 지표의 버퍼 수는 컴파일러 지시문을 사용하여 미리 선언해야하는데, 예를 들면 :

```
#property indicator_buffers 3 // 버퍼의 수
#property indicator_plots   2 // 프롯의 수
```

지표의 버퍼 수는 항상 지표의 플롯 수보다 크거나 같습니다.

각 기본 플롯 타입은 색상 변화 또는 구성상 특이사항이 있을 수 있으므로 MQL5의 실제 플롯 타입은 18개입니다:

플로팅	설명	값 버퍼	색상 버퍼
DRAW_NONE	차트에 시각적으로 표시되지 않지만 해당 버퍼 값은 데이터 창에서 볼 수 있습니다	1	-
DRAW_LINE	해당 버퍼의 값에 선이 표시됩니다(버퍼의 빈 값	1	-

© 2000-2025, MetaQuotes Ltd.

플로팅	설명	값 버퍼	색상 버퍼
	은 바람직하지 않음)		
DRAW_SECTION	해당 버퍼 값 사이의 선 세그먼트로 그려집니다 (일반적으로 빈 값이 많음)	1	-
DRAW_HISTOGRAM	0 라인에서 해당 버퍼 값까지 히스토그램으로 그려집니다 (빈 값이 있을 수 있음)	1	-
DRAW_HISTOGRAM2	두 지표 버퍼를 기준으로 히스토그램으로 그려집니다(빈 값이 있을 수 있음)	2	-
DRAW_ARROW	심볼로 그려집니다(빈 값이 있을 수 있음)	1	-
DRAW_ZIGZAG	DRAW_SECTION 스타일과 유사하지만, 이와 달리 수직 세그먼트를 하나의	2	-

© 2000-2025, MetaQuotes Ltd.

플로팅	설명	값 버퍼	색상 버퍼
	막대에 플롯할 수 있습니다		
DRAW_FILLING	두 줄 사이의 색 채우기 해당 버퍼의 값 2개가 데이터 창에 표시됩니다.	2	-
DRAW_BARS	막대로 그려집니다. 해당 버퍼의 4개의 값이 데이터 창에 표시됩니다	4	-
DRAW_CANDLES	일본 캔들스틱으로 그려집니다. 해당 버퍼의 4개의 값이 데이터 창에 표시됩니다	4	-
DRAW_COLOR_LINE	언제든지 서로 다른 막대의 색상을 바꾸거나 색상을 변경할 수 있는 선	1	1
DRAW_COLOR_SECTION	DRAW_SECTION 스타일과 유사하지만 각 섹	1	1

© 2000-2025, MetaQuotes Ltd.

플로팅	설명	값 버퍼	색상 버퍼
	션의 색상을 개별적으로 설정할 수 있으며, 색상을 동적으로 설정할 수도 있습니다		
DRAW_COLOR_HISTOGRAM	DRAW_HISTOGRAM스타일과 유사하지만 각 스트립의 색상이 다를 수 있으므로 색상을 동적으로 설정할 수 있습니다	1	1
DRAW_COLOR_HISTOGRAM2	DRAW_HISTOGRAM2 스타일과 유사하지만 각 스트립의 색상이 다를 수 있으므로 색상을 동적으로 설정할 수 있습니다	2	1
DRAW_COLOR_ARROW	DRAW_ARROW스타일과 유사하지만 각 심볼은 색	1	1

© 2000-2025, MetaQuotes Ltd.

플로팅	설명	값 버퍼	색상 버퍼
	상을 가질 수 있습니다. 동적으로 색상을 변경할 수 있습니다		
DRAW_COLOR_ZIGZAG	DRAW_ZIGZAG 스타일로 섹션 컬러링 옵션과 동적 색상 전환 옵션이 제공됩니다	2	1
DRAW_COLOR_BARS	DRAW_BARS 스타일로 막대의 개별 컬러링 옵션과 동적 색상 변경 옵션이 제공됩니다	4	1
DRAW_COLOR_CANDLES	DRAW_CANDLES 스타일로 캔들스틱의 개별 컬러링 옵션과 동적 색상 변경 옵션이 제공됩니다	4	1

지표 버퍼와 배열의 차이

© 2000-2025, MetaQuotes Ltd.

각 지표의 글로벌 레벨에서, double 유형의 배열을 하나 이상 선언해야 하며, 이 배열은 SetIndexBuffer() 기능을 사용하여 지표 버퍼로 사용해야 합니다. 지표 플롯을 그리려면 지표 버퍼의 값만 사용되며, 이 용도로는 다른 배열을 사용할 수 없습니다. 또한 버퍼 값이 데이터 창에 표시됩니다.

지표 버퍼는 동적이어야 하며 크기를 지정할 필요가 없으며, 지표 버퍼로 사용되는 배열의 크기는 터미널 실행 하위 시스템에 의해 자동으로 설정됩니다.

배열이 지표 버퍼에 바인딩된 후에는, 인덱싱 방향이 일반 배열과 같이 기본적으로 설정되지만 ArraySetAsSeries() 기능을 사용하여 배열 요소에 대한 액세스 방법을 변경할 수 있습니다. 기본적으로 지표 버퍼는 플로팅에 사용되는 데이터를 저장하는 데 사용됩니다(INDICATOR_DATA).

지표 값을 계산하기 위해 중간 계산을 보류하고 각 막대에 대한 추가 값을 저장해야 하는 경우, 바인딩 중에 이러한 배열을 계산 버퍼로 선언할 수 있습니다 (INDICATOR_CALCULATIONS). 중간 값의 경우 일반 배열도 사용할 수 있지만 이 경우에는 프로그래머가 배열 크기를 관리해야 합니다.

일부 플롯에서는 각 막대의 색상을 설정할 수 있습니다. 색에 대한 정보를 저장하려면 색상 버퍼를 사용합니다 (INDICATOR_COLOR_INDEX). 색상은 정수 유형 색상이지, 모든 지표 버퍼는 double 유형이어야 합니다. CopyBuffer()를 사용하여 색상 및 보조 (INDICATOR_CALCULATIONS) 버퍼 값을 가져올 수 없습니다.

지표 버퍼 수는 컴파일러 지시문 #property indicator_buffers number_of_buffers를 사용하여 지정해야 합니다:

```
#property indicator_buffers 3  //  지표는 3개의 버퍼가 있습니다
```

하나의 지표에 허용되는 최대 버퍼 수는 512개입니다.

지표 버퍼 및 플롯의 연관성

각 플롯은 하나 이상의 지표 버퍼를 기반으로 합니다. 따라서 간단한 캔들스틱을 표시하기 위해서는 시가, 고가, 저가, 종가의 네 가지 값이 필요합니다. 따라서 캔들스틱 형태의 지표를 표시하려면 지표 버퍼 4개와 double 유형의 배열 4개를 선언해야 합니다. 예를 들어:

```
//--- 지표에는 4개의 지표 버퍼가 있습니다
#property indicator_buffers 4
//--- 지표에는 하나의 플로팅이 있습니다
#property indicator_plots   1
//--- 그래픽 플로팅 1번이 캔들스틱으로 나타납니다
#property indicator_type1   DRAW_CANDLES
//--- 캔들스틱은 clrDodgerBlue로 그려집니다
#property indicator_color1  clrDodgerBlue
//--- 지표 버퍼를 위한 4개의 배열
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

그래픽 플롯은 플롯 번오에 따라 지표 버퍼를 자동으로 사용합니다. 플롯 번호 지정은 1로 시작하고 버퍼 번호 지정은 0으로 시작합니다. 첫 번째 플로팅에 지표 버퍼 4개가 필요한 경우, 지표 버퍼 4개를 사용하

© 2000-2025, MetaQuotes Ltd.

여 지표 버퍼를 그립니다. 이러한 4개 퍼버는 SetIndexBuffer() 기능을 사용하여 올바른 인덱싱을 가진 배열과 연결해야 합니다.

```
//--- 지표 버퍼로 배열 바인딩
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // 첫 번째 버퍼는 0 인덱스에 해당합니다
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // 두 번째 버퍼는 1 인덱스에 해당합니다
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // 세 번째 버퍼는 2 인덱스에 해당합니다
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // 네 번째 버퍼는 3 인덱스에 해당합니다
```

"캔들스틱"의 플로팅이 첫 번째 번호로 발표되었기 때문에, 지표인 캔들스틱은 첫 번째 4개의 버퍼만 사용합니다.

예를 변경하고 간단한 선의 플로팅을 추가하는 – DRAW_LINE. 이제 선 번호가 1이고 캔들스틱이 2라고 가정합니다. 버퍼 수와 플롯 수가 증가했습니다.

```
//--- 지표에는 5개의 지표 버퍼가 있습니다
#property indicator_buffers 5
//--- 지표에는 2개의 플롯이 있습니다
#property indicator_plots   2
//--- 플롯 1은 선입니다
#property indicator_type1   DRAW_LINE
//--- 선의 색상은 clrDodgerRed 입니다
#property indicator_color1  clrDodgerRed
//--- 플롯 2는 일본 캔들스틱으로 그려집니다
#property indicator_type2   DRAW_CANDLES
//--- 캔들스틱의 색은 clrDodgerBlue입니다
#property indicator_color2  clrDodgerBlue
//--- 지표 버퍼를 위한 5개의 배열
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

변경된 플롯의 순서가 바뀌어서 이제 선이 먼저 나오고, 이어서 일본 캔들스틱이 나옵니다. 따라서 버퍼 순서가 적절하며, 먼저 인덱스가 0인 선에 버퍼를 발표한 다음 캔들스틱에 버퍼를 4개 지정합니다.

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // The first buffer corresponds to in
//--- 캔들스틱에 대한 지표 버퍼로 배열 바인딩
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // 두 번째 버퍼는 1 인덱스에 해당합니다
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // 세 번째 버퍼는 2 인덱스에 해당합니다
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // 네 번째 버퍼는 3 인덱스에 해당합니다
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);     // 다섯 번째 버퍼는 4 인덱스에 해당합니다
```

버퍼와 플롯의 수는 컴파일러 지시문을 사용해서만 설정할 수 있으며, 함수를 사용해서 이러한 속성을 동적으로 변경할 수는 없습니다.

스타일의 색상 버전

© 2000-2025, MetaQuotes Ltd.

표에서 볼 수 있듯이 스타일은 두 그룹으로 나뉩니다. 첫 번째 그룹은 이름에 **COLOR** 단어가 없는 스타일을 포함하며, 이러한 스타일을 기본 스타일이라고 합니다:

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

두 번째 그룹은 이름에 **COLOR**단어를 포함하는 스타일이며, 이를 색상 버전이라고 합니다:

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM
- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

모든 색상 버전은 플롯의 각 부분에 색상을 지정할 수 있다는 점에서 기본 버전과 다릅니다. 플로팅의 최소 부분은 막대이므로 각 막대에 색상을 설정할 수 있습니다.

색상 버전에 없는 DRAW_NONE 및 DRAW_FILLING, 스타일은 예외입니다.

각 막대에 플로팅 색상을 설정하기 위해 색상 인덱스을 저장하기 위한 버퍼가 색상 버전에 추가되었습니다. 이러한 인덱스는 미리 정의된 색상 세트를 포함하는 특수 배열의 색상 수를 나타냅니다. 색상 배열의 크기는 64입니다. 즉, 스타일의 각 색상 버전은 64가지 색상으로 플롯을 그릴 수 있습니다.

필요한 모든 색상을 쉼표로 구분하여 지정할 수 있는 컴파일러 지시문 #property indicator_color를 통해 색상 세트와 섹상을 설정할 수 있습니다. 예를 들어 지표에 있는 이러한 항목은 다음과 같습니다:

```
//--- 컬러링 캔들스틱에 8가지 색상을 정의(특수 배열에 저장됨)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
```

플롯 1, 8가지 색상이 설정되며, 이는 특수 배열에 배치됩니다. 또한 이 프로그램에서는 플롯의 색상은 지정하지 않고 인덱스만 지정합니다. 막대 번호K에 대해 빨강색으로 설정하려면 배열의 색상 인덱스 값ㅇ을 지표의 색상 버퍼에 설정해야 합니다. 빨강색은 지시문에 먼저 지정되며 인덱스 번호 0에 해당합니다.

```
  //--- 캔들스틱 색상을 clrRed로 설정
  col_buffer[buffer_index]=0;
```

색상 세트는 한 번 제공되는 것이 아니며, PlotIndexSetInteger()를 사용하여 동적으로 변경할 수 있습니다. 예:

```
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
```

© 2000-2025, MetaQuotes Ltd.

```
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
                          color_array[i]);      //  새로운 색상
```

지표 및 플로팅의 속성

지표 플롯의 경우 컴파일러 지시문과 적절한 함수를 사용하여 속성을 설정할 수 있습니다. 지표 속성 및 기능 간의 연결에서 이에 대한 자세한 정보 읽기. 특수 기능을 사용하여 지표 속성을 동적으로 변경하면 보다 유연한 사용자 정의 지표를 만들 수 있습니다.

차트에 지표 그리기 시작

대부분의 경우, 알고리즘 조건에 따라 내역에서 사용할 수 있는 최소 개수의 이전 막대를 제공해야 하므로 현재 막대로 지표 값 계산을 즉시 시작할 수 없습니다. 예를 들어, 많은 유형의 스무딩은 이전 N 막대에 대한 가격 배열을 사용하는 것을 의미하며, 이러한 값을 기반으로 현재 막대의 지표 값이 계산됩니다.

이러한 경우, 첫 번째 N 막대에 대한 지표 값을 계산할 방법이 없거나, 이러한 값은 차트에 표시되도록 의도한 것이 아니며 추가 값을 계산하기 위한 보조적인 것일 뿐입니다. 히스토리의 첫 N 막대에 지표가 플로팅되지 않도록 하려면 N값을 해당 플롯의 PLOT_DRAW_BEGIN 속성으로 설정합니다:

```
//--- 캔들스틱에 대한 지표 버퍼로 배열 바인딩
PlotIndexSetInteger(number_of_plot,PLOT_DRAW_BEGIN,N);
```

이곳:

- number_of_plot – 0에서 indicator_plots-1까지의 값 (플롯의 합은 0으로 시작됨).
- N – 차트에 표시해서는 안 되는 내역의 첫 번째 막대 수.

© 2000-2025, MetaQuotes Ltd.

DRAW_NONE

DRAW_NONE 스타일은 버퍼의 값을 계산하여 데이터 창에 표시해야 하는 경우에 사용하도록 설계되었지만 차트의 플롯은 필요하지 않습니다. 정확도를 설정하려면 OnInit() 함수에서 IndicatorSetInteger(INDICATOR_DIGITS,num_chars를 사용합니다:

```
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- 데이터 창에 표시할 값의 정확도 설정
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

DRAW_NONE을 플로팅하는 데 필요한 버퍼 수는 1개입니다.

데이터 창에서 마우스가 현재 가리키는 막대의 수를 보여주는 지표의 예. 번호 지정은 시계열과 일치하며, 즉, 현재 완료되지 않은 막대의 인덱스가 0이고 가장 오래된 막대의 인덱스가 가장 큽니다.

빨간색은 플로팅 **#1**로 설정되어 있지만 지표는 차트에 아무 것도 그리지 않습니다.

```
//+------------------------------------------------------------------+
//|                                                    DRAW_NONE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Invisible 플롯
#property indicator_label1  "막대 인덱스"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- 지표 버퍼
double         InvisibleBuffer[];
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 및 배열 바인딩
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- 데이터 창에 표시할 값의 정확도 설정
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static datetime lastbar=0;
//--- 지표의 첫 번째 계산인 경우
   if(prev_calculated==0)
     {
      //--- 막대 번호를 처음으로 다시 매깁니다
      CalcValues(rates_total,close);
      //--- 마지막 막대에서 현재 막대의 개방 시간을 기억
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
     }
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 새 막대가 나타난 경우 막대와 개방 시간이 다릅니다
      if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
        {
         //--- 막대 번호를 다시 한 번 변경
         CalcValues(rates_total,close);
         //--- 마지막 막대에 있는 현재 막대의 개방 시간 업데이트
         lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 시계열처럼 막대에 번호를 매김                                    |
//+------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- 지표 버퍼의 인덱싱을 시계열처럼 설정
   ArraySetAsSeries(InvisibleBuffer,true);
//--- 각 막대에 해당 번호를 입력
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

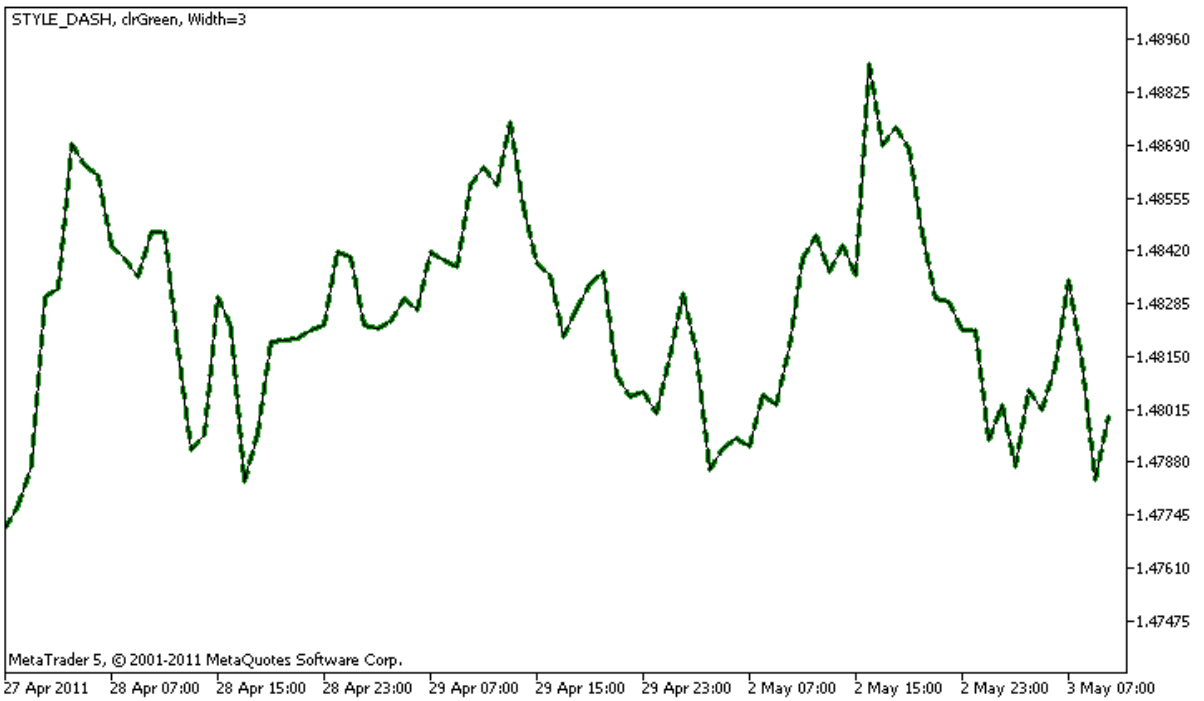
© 2000-2025, MetaQuotes Ltd.

DRAW_LINE

DRAW_LINE은 지표 버퍼 값을 기준으로 지정된 색상의 선을 그립니다. 선의 폭, 스타일 및 색상은 컴파일러 지시문과 PlotIndexSetInteger() 기능을 사용하여 동적으로 설정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

DRAW_LINE을 플로팅하는 데 필요한 버퍼 수는 1개입니다.

막대의 종가를 사용하여 선을 그리는 지표의 예. N=5 틱마다 선 색상, 너비 및 스타일이 임의로 변경됩니다.



처음에는 DRAW_LINE의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 속성을 설정한 다음, OnCalculate() 기능에서 이 세 속성을 임의로 설정합니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_LINE을 시연하기 위한 지표"
#property description "종가에서 지정된 색상의 선을 그립니다"
#property description "선의 색상, 너비 및 스타일이 임의로 변경됩니다"
#property description "모든 N 틱 후"

#property indicator_chart_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- 선 속성은 컴파일러 지시문을 사용하여 설정합니다
#property indicator_label1  "Line"      // 데이터 창의의 플롯 이름
#property indicator_type1   DRAW_LINE   // 플로팅 유형은 선입니다
#property indicator_color1  clrRed      // 선 색상
#property indicator_style1  STYLE_SOLID // 선 스타일
#property indicator_width1  1           // 선 너비
//--- 매개변수 입력
input int      N=5;         // 변경할 틱의 수
//--- 플롯에 대한 지표 버퍼
double         LineBuffer[];
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen};
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 및 배열 바인딩
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- 모조 난수 생성기 초기화
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
```

© 2000-2025, MetaQuotes Ltd.

```
      ChangeLineAppearance();
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산을 위한 블록
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- 함수의 다음 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지표에 그려진 선 모양을 변경                                     |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선의 색상을 변경하기 위한 블록
//--- 난수 가져오기
   int number=MathRand();
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);
//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 선 색상 쓰기
   comm=comm+(string)colors[color_index];

//--- 선 너비 변경 블록
   number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5; // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+", Width="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm=EnumToString(styles[style_index])+", "+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_SECTION

DRAW_SECTION 지표 버퍼 값을 기준으로 지정된 색상의 섹션을 그립니다. 라인의 너비, 색상 및 스타일은 DRAW_LINE 스타일과 마찬가지로 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

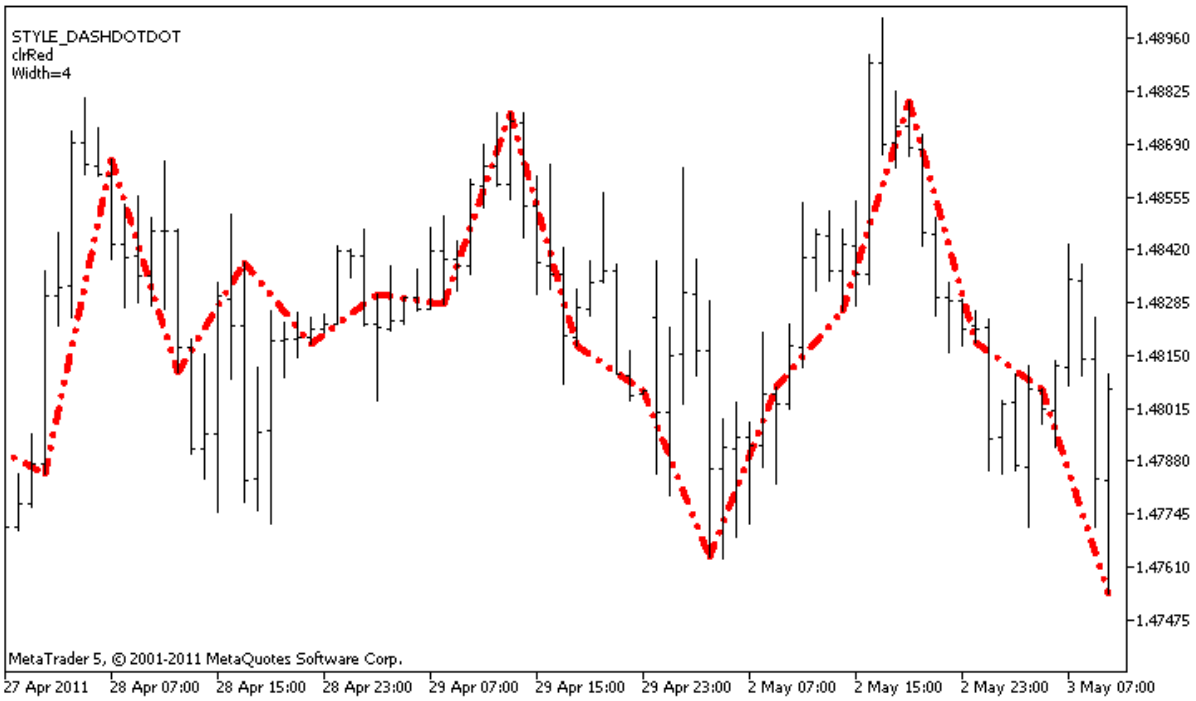
섹션이 비어있지 않은 한 값에서 비어 있지 않은 다른 표시 버퍼 값으로 그려지며, 비어 있는 값은 무시됩니다. "빈 값"으로 간주할 값을 지정하려면 PLOT_EMPTY_VALUE 속성에 이 값을 설정: 예를 들어 0이 아닌 값에 대해 일련의 섹션으로 지표를 그려야 하는 경우, 0값을 빈 값으로 설정해야 합니다:

```
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(index_of_plot_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

지표 버퍼의 값을 항상 명시적으로 채우고 버퍼의 빈 값을 표시해서는 안 되는 요소로 설정하십시오.

DRAW_SECTION을 플로팅하는 데 필요한 버퍼 수는 1개입니다.

고가 및 저가 사이에 섹션을 그리는 지표의 예. **모든** 섹션의 색상, 너비 및 스타일은 **N** 틱을 누를 때마다 임의로 변경됩니다.



처음에는 DRAW_SECTION의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 속성을 설정한 다음, OnCalculate() 기능에서 이 세 속성을 임의로 설정합니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                                DRAW_SECTION.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"

#property description "DRAW_SECTION을 시연하기 위한 지표"
#property description "모든 막대에 직선 섹션 그리기"
#property description "섹션의 색상, 너비 및 스타일이 임의로 변경됩니다"
#property description "모든 N 틱 후"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Section 플롯
#property indicator_label1  "섹션"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      bars=5;           // 막대의 섹션 길이
input int      N=5;              // 섹션 스타일을 변경할 틱 수
//--- 플롯에 대한 지표 버퍼
double         SectionBuffer[];
//--- 섹션의 끝을 계산하는 보조 변수
int            divider;
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen};
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 및 배열 바인딩
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 지표 매개변수 확인
   if(bars<=0)
     {
      PrintFormat("유효하지 않은 매개변수 막대의 값=%d",bars);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
//---+
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산이 시작되는 막대의 수
   int start=0;
//--- 지표가 이전에 계산된 적이 있는 경우 이전 막대에서 시작을 설정하십시오
   if(prev_calculated>0) start=prev_calculated-1;
//--- 다음은 지표 값의 모든 계산값입니다
   for(int i=start;i<rates_total;i++)
     {
      //--- 막대 번호의 나머지 2*막대를 가져오기
      int rest=i%divider;
      //--- 막대 번호를 2*막대로 나눈 경우
      if(rest==0)
        {
         //--- 섹션 끝을 이 막대의 고가로 설정
         SectionBuffer[i]=high[i];
        }
      //---나눗셈의 나머지가 막대와 같은 경우,
      else
        {
         //--- 섹션 끝을 이 막대의 고가로 설정
         if(rest==bars) SectionBuffer[i]=low[i];
         //--- 아무 일도 없는 경우, 막대 세트 0을 무시
         else SectionBuffer[i]=0;
        }
     }
//--- 함수의 다음 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 지표의 섹션 모양을 변경                                          |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선 색상 변경 블록
   int number=MathRand(); // 임의 숫자 가져오기
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);
//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 선 색상 쓰기
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 선 너비 변경 블록
   number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5;   // 너비는 0 ~ 4로 설정됩니다
//--- 너비 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 선 스타일 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

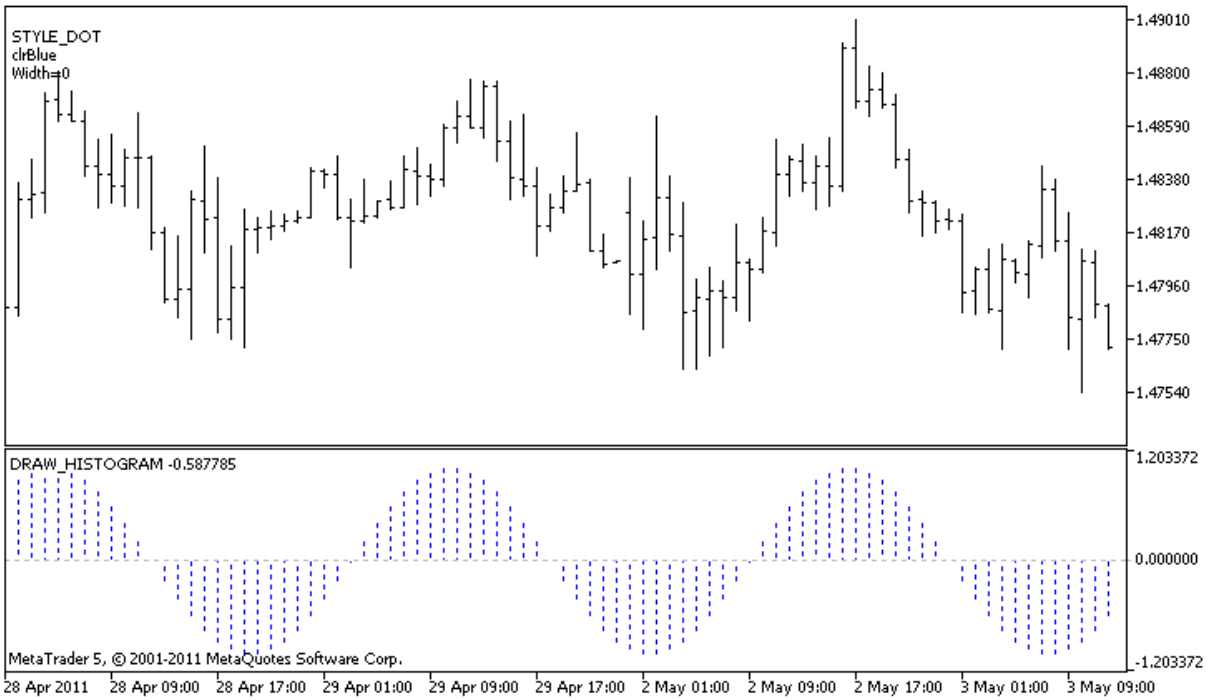
DRAW_HISTOGRAM

DRAW_HISTOGRAM 스타일은 0부터 지정된 값까지 지정된 색상의 열 순서로 히스토그램을 그립니다. 값은 지표 버퍼에서 가져옵니다. 열의 너비, 색상 및 스타일은 DRAW_LINE 스타일과 마찬가지로 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플롯 속성을 동적으로 변경하여 현재 상황에 따라 히스토그램의 모양을 변경할 수 있습니다.

각 막대에 0 레벨의 열이 그려지므로 DRAW_HISTOGRAM을 별도의 차트 창에 사용하는 것이 좋습니다. 이러한 유형의 플롯은 대개 오실레이터 유형의 지표, 예를 들면 Bears Power 또는 OsMA를 생성하는 데 사용됩니다. 표시할 수 없는 빈 값의 경우 0 값을 지정해야 합니다.

DRAW_HISTOGRAM을 플로팅하는 데 필요한 버퍼의 수는 1개입니다.

지정된 색상의 사인군을 MathSin() 함수 기반으로 그리는 지표의 예. **모든** 히스토그램 열의 색상, 너비 및 스타일이 각 **N** 틱으로 임의 변경됩니다. 막대 매개변수는 사인파의 주기를 지정하며, 이 주기는 지정된 막대의 수 이후이며, 사인파는 사이클을 반복합니다.



처음에는 DRAW_HISTOGRAM의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 속성을 설정한 다음 OnCalculate() 기능에서 이 세 속성을 임의로 설정합니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_HISTOGRAM을 시연하기 위한 지표"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "별도의 창에 사인파를 히스토그램으로 그립니다"
#property description "열의 색상과 너비가 임의로 변경됩니다"
#property description "모든 N 틱 후"
#property description "막대 매개변수는 사인파 사이클의 막대 수를 설정합니다"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Histogram 플롯
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      bars=30;          // 막대의 사인파 주기
input int      N=5;              // 히스토그램을 변경할 틱
//--- 지표 버퍼
double         HistogramBuffer[];
//--- 막대 매개변수를 곱산 경우 라디안 단위로 2Pi 각도를 구하는 계수
double    multiplier;
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen};
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- 승수를 계산
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Set the value of bars=%d greater than 1",bars);
      //--- 지표의 조기 종료
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산
   int start=0;
//--- 이전 OnCalculate 시작 중에 이미 계산된 경우
   if(prev_calculated>0) start=prev_calculated-1; // 계산의 시작을 단 하나의 막대로 설정
//--- 지표 버퍼에 값을 입력
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- 함수의 다음 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지표에 있는 선의 모양을 변경                                     |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선의 색상을 변경하기 위한 블록
   int number=MathRand(); // 임의 숫자 가져오기
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);
//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 선 색상 쓰기
   comm=comm+"\r\n"+(string)colors[color_index];
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 선 너비 변경 블록
   number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5;   // 너비는 0 ~ 4로 설정됩니다
//--- 너비 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 선 스타일 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_HISTOGRAM2

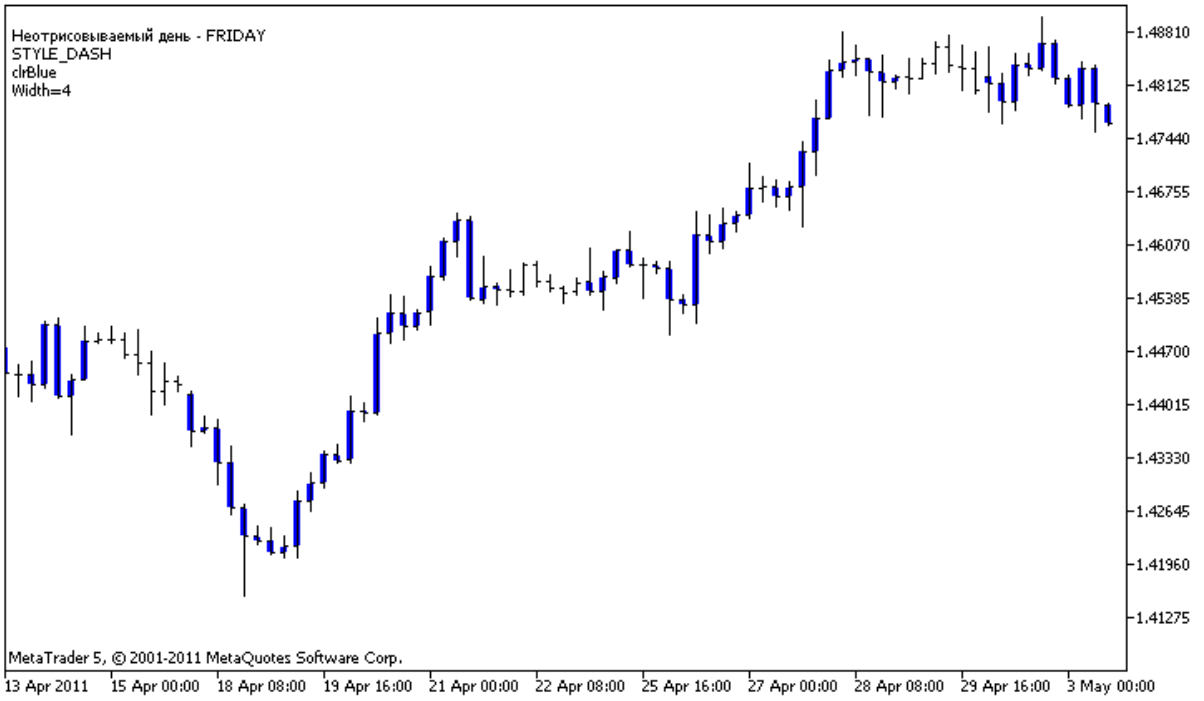
DRAW_HISTOGRAM2 스타일은 두 지표 버퍼 값을 사용하여 지정된 색상(수직 세그먼트)의 히스토그램을 그립니다. 세그먼트의 너비, 색상 및 스타일은 DRAW_LINE 스타일과 마찬가지 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플롯 속성을 동적으로 변경하여 현재 상황에 따라 히스토그램의 모양을 변경할 수 있습니다.

DRAW_HISTOGRAM2 스타일은 차트의 별도의 하위 창과 기본 창에서 사용할 수 있습니다. 빈 값은 아무것도 그려지지 않으므로 지표 버퍼의 모든 값을 명시적으로 설정해야 합니다. 버퍼는 0 값으로 초기화되지 않습니다.

DRAW_HISTOGRAM2를 플로팅하는 데 필요한 버퍼의 수는 2개입니다.

각 막대의 시가 및 종가 사이에서 지정된 색상과 너비의 수직 세그먼트를 플롯하는 지표의 예. 모든 히스토그램의 열의 색상, 너비 및 스타일이 각 N 틱으로 임의 변경됩니다. 지표를 시작하지 않는 동안, OnInit() 기능에서 히스토그램이 그려지지 않은 요일의 수(invisible_day)는 임의로 설정됩니다. 이를 위해 빈 값이 설정됩니다. PLOT_EMPTY_VALUE=0:

```
//--- 빈 값 설정
   PlotIndexSetDouble(index_of_plot_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```



처음에는 DRAW_HISTOGRAM2의 **plot1**의 경우 컴파일러 지시문 #property을 사용하여 속성을 설정한 다음, OnCalculate() 기능에서 이 세 속성을 임의로 설정합니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM2.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
```

© 2000-2025, MetaQuotes Ltd.

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_HISTOGRAM2를 시연하기 위한 지표"
#property description "각 막대의 시가 및 종가 사이에 세그먼트를 그립니다"
#property description "색상, 너비 및 스타일이 임의로 변경됩니다"
#property description "모든 N 틱 후"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- Histogram_2 플롯
#property indicator_label1  "Histogram_2"
#property indicator_type1   DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      N=5;              // 히스토그램을 변경할 틱
//--- 지표 버퍼
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- 지표가 플롯되지 않은 요일
int invisible_day;
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen};
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- 빈 값 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 0 ~ 5 사이의 난수 가져오기
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산
   int start=0;
//--- 각 막대의 시가별 요일을 가져오기 위해
   MqlDateTime dt;
//--- 이전 OnCalculate 시작 중에 이미 계산된 경우
   if(prev_calculated>0) start=prev_calculated-1; // 계산의 시작을 단 하나의 막대로 설정
//--- 지표 버퍼에 값을 입력
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      if(dt.day_of_week==invisible_day)
        {
         Histogram_2Buffer1[i]=0;
         Histogram_2Buffer2[i]=0;
        }
      else
        {
         Histogram_2Buffer1[i]=open[i];
         Histogram_2Buffer2[i]=close[i];
        }
     }
//--- 함수의 다음 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지표에 있는 선의 모양을 변경                                     |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선 색상 변경 블록
   int number=MathRand(); // 임의 숫자 가져오기
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);
//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 선 색상 쓰기
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 선 너비 변경 블록
   number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5;   // 너비는 0 ~ 4로 설정됩니다
//--- 선 너비 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 선 스타일 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- 계산에 생략된 날짜에 대한 정보 추가
   comm="\r\nNot plotted day - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_ARROW

The DRAW_ARROW 스타일은 지표 버퍼 값을 기준으로 지정된 색상(설정된 Wingdings의 심볼)의 화살표를 그립니다. 기호의 폭과 색상은 DRAW_LINE 스타일처럼 지정할 수 있으며 컴파일러 지시문을 사용하거나PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플로팅 특성의 동적 변화를 통해 현재 상황에 따라 지표의 모양을 변경할 수 있습니다.

심볼 코드는 PLOT_ARROW 속성을 사용해서 설정됩니다.

```
//--- PLOT_ARROW에 Wingdings 폰트에서 심볼을 정의
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

PLOT_ARROW=159 의 기본값 (원).

각 화살표는 실제로 높이와 고정점이 있는 기호이며 차트에서 몇 가지 중요한 정보(예: 막대의 종가)를 포함할 수 있습니다. 따라서 차트의 척도에 따라 달라지지 않는 픽셀 단위로 수직 이동을 추가로 지정할 수 있습니다. 화살표는 지정된 픽셀 수만큼 아래로 이동하지만 지표 값은 동일하게 유지됩니다:

```
//--- 화살표의 수직 이동 설정(픽셀)
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

PLOT_ARROW_SHIFT의 음수 값은 위쪽 화살표 이동을 의미하며, 양수 값은 아래쪽 화살표 이동을 의미합니다.

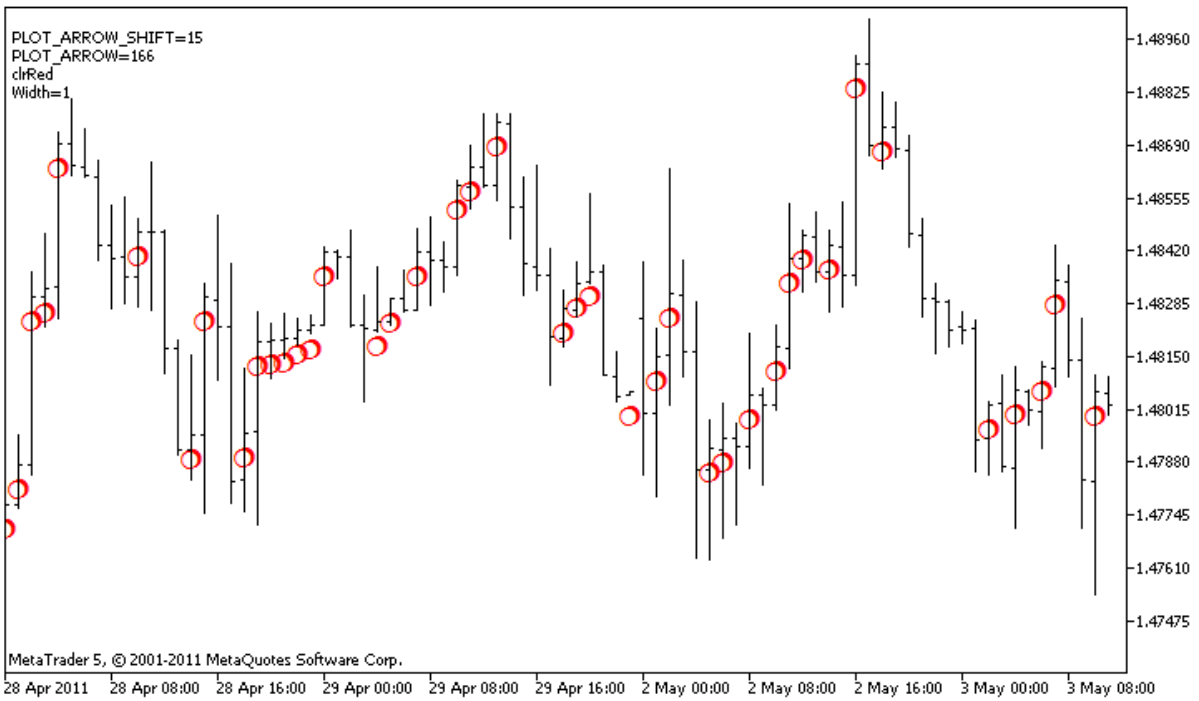
DRAW_ARROW 스타일은 차트의 별도 하위 창과 기본 창에서 사용할 수 있습니다. 빈 값은 그려지지 않으며 "데이터 창"에 나타나지 않으니, 지표 버퍼의 모든 값을 명시적으로 설정해야 합니다. 버퍼는 0 값으로 초기화되지 않습니다.

```
//--- 빈 값 설정
   PlotIndexSetDouble(index_of_plot_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

DRAW_ARROW 플로팅하는 데 필요한 버퍼 수는 1개 입니다.

지표의 예로, 각 막대에 이전 막대의 종가보다 높은 가격으로 화살표를 그립니다. 모든 화살표의 색상, 너비, 이동 및 기호 코드가 N 틱마다 임의로 변경됩니다.

© 2000-2025, MetaQuotes Ltd.



예제에서 DRAW_ARROW 스타일을 사용하는 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 속성, 색상 및 크기를 지정한 다음 OnCalculate() 함수에서 속성을 임의로 설정합니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ARROW.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_ARROW를 시연하기 위한 지표"
#property description "유니코드 문자로 설정된 화살표 차트를 그림"
#property description "화살표의 색상, 크기, 이동 및 심볼 코드가 임의로 변경됩니다"
#property description "모든 N 틱 후"
#property description "코드 매개변수는 기준 값을 설정: code=159 (a circle)"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- 화살표를 플롯
#property indicator_label1  "화살표"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- 매개변수 입력
input int      N=5;          // 변경할 틱의 수
input ushort   code=159;     // DRAW_ARROW에 그릴 심볼 코드
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 플롯에 대한 지표 버퍼
double         ArrowsBuffer[];
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- PLOT_ARROW에서 도면에 대한 심볼 코드 정의
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- 화살표의 수직 이동 설정(픽셀)
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- 빈 값 0으로 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 화살표의 색상, 크기, 이동 및 코드를 변경하는 틱 계산
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산을 위한 블록
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 계산 루프
   for(int i=1;i<rates_total;i++)
     {
      //--- 현재 종가가 이전 가격보다 높은 경우 화살표를 그립니다
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- 그렇지 않으면 0값을 지정
      else
         ArrowsBuffer[i]=0;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지표의 심볼 모양 변경                                            |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 지표 속성에 대한 정보 형성을 위한 문자열
   string comm="";
//--- 화살표 색상을 변경하는 블록
   int number=MathRand(); // 임의 숫자 가져오기
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);
//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 선 색상 쓰기
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 화살표 크기를 변경할 수 있는 경우
   number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5;   // The size is set from 0 to 4
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 배열 크기 쓰기
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 화살표 코드 변경 블록(PLOT_ARROW)
   number=MathRand();
//--- 화살표의 새 코드(0 ~ 19)를 계산할 수 있는 정수 분할의 나머지를 가져오기
   int code_add=number%20;
//--- code+code_add의 결과로 새 기호 코드를 설정
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- 심볼 코드 PLOT_ARROW 쓰기
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 화살표의 수직 이동(픽셀)을 변경하기 위한 블록
   number=MathRand();
//--- 이동을 정수 나눗셈의 나머지 값으로 가져오기
   int shift=20-number%41;
//--- 새 이동을 -20 ~ 20으로 설정
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- 이동 PLOT_ARROW_SHIFT 쓰기
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_ZIGZAG

DRAW_ZIGZAG 스타일은 두 지표 버퍼 값을 기반으로 지정된 색상의 세그먼트를 그립니다. 이 스타일은 DRAW_SECTION과 매우 유사하지만, 두 지표버퍼 값이 모두 이 막대에 설정된 경우 한 막대 내에 수직 세그먼트를 그릴 수 있습니다. 세그먼트의 첫 번째 버퍼의 값에서 두 번째 지표 버퍼의 값으로 표시됩니다. 이 경우에는 아무것도 표시되지 않으므로 어떤 버퍼도 빈 값만 포함할 수 없습니다.

선의 너비, 색상 및 스타일은 DRAW_SECTION 스타일과 마찬가지로 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

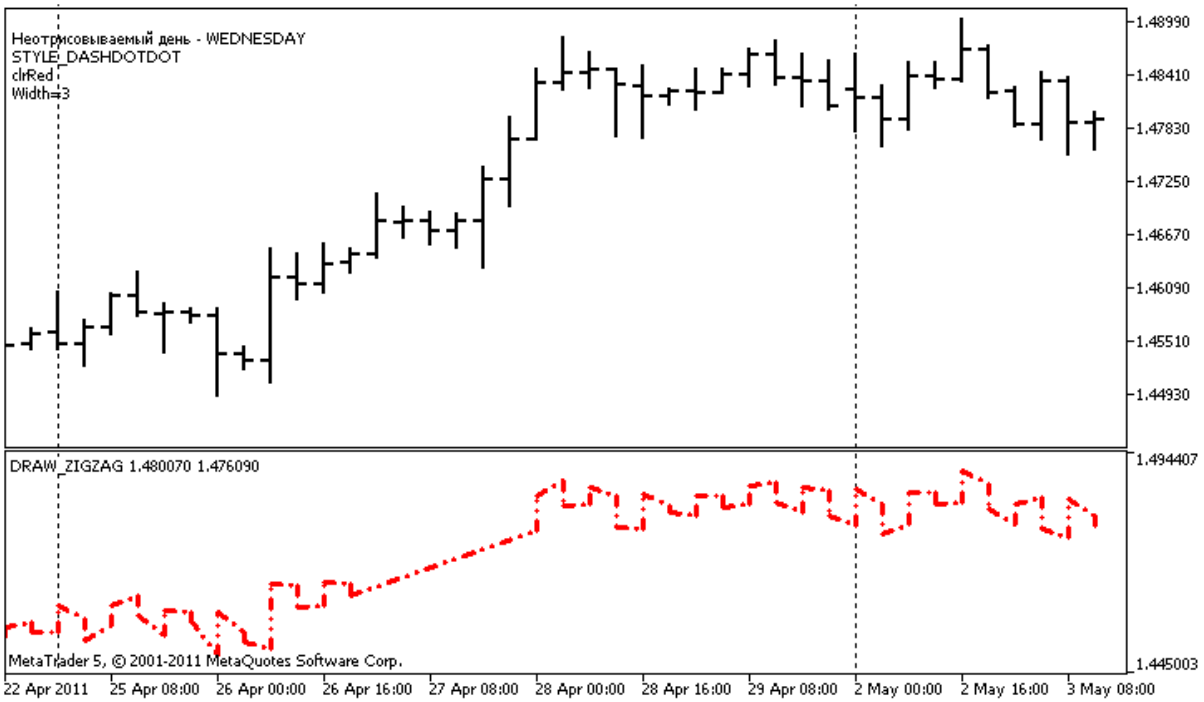
섹션은 한 버퍼의 비어 있지 않은 값에서 다른 지표 버퍼의 비어 있지 않은 값으로 그려집니다. "빈 값"으로 간주할 값을 지정하려면 PLOT_EMPTY_VALUE 속성에 이 값을 설정:

```
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(index_of_plot_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

항상 지표 버퍼의 값을 명시적으로 입력하고 버퍼에서 빈 값을 건너뛰도록 설정합니다.

DRAW_ZIGZAG를 플로팅에 필요한 버퍼의 수는 2개입니다.

고가와 저가를 기준으로 지그재그를 표시하는 지표의 예. 지그재그 선의 색상, 너비 및 스타일은 N 틱마다 임의로 변경됩니다.



처음에는 DRAW_ZIGZAG의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 속성을 설정한 다음, OnCalculate() 기능에서 임의로 설정합니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                                  DRAW_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_ZIGZAG를 시연하기 위한 지표"
#property description "하루의 막대를 건너뛰고 ₩saw₩ 을 직선 세그먼트로 그립니다"
#property description "건너뛸 요일은 지표 시작 중이 임의로 선택됩니다"
#property description "세그먼트의 색상, 너비 및 스타일이 임의로 변경됩니다"
#property description " 모든 N 틱"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- ZigZag 플롯
#property indicator_label1  "ZigZag"
#property indicator_type1   DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      N=5;              // 변경할 틱의 수
//--- 지표 버퍼
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- 지표가 플롯되지 않은 요일
int invisible_day;
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen};
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 배열 및 지표 버퍼 바인딩
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- 0 ~ 6사이의 임의 값을 가져오고, 이 날의 지표는 표시되지 않습니다
   invisible_day=MathRand()%6;
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 사용자 지정 지표 반복 함수                                        |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 충분한 수의 틱이 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 시간 구조는 각 막대의 요일을 가져오는 데 필요합니다
   MqlDateTime dt;

//--- 계산 시작 위치
   int start=0;
//--- 지표가 이전 틱에서 계산된 경우, 마지막 틱에서 한 틱으로 계산을 시작합니다
   if(prev_calculated!=0) start=prev_calculated-1;
//--- 계산 루프
   for(int i=start;i<rates_total;i++)
     {
      //--- 구조에 막대 오픈 시간 기록
      TimeToStruct(time[i],dt);
      //--- 이 막대의 요일이 invisible_day와 같은 경우
      if(dt.day_of_week==invisible_day)
        {
         //--- 이 막대의 버퍼에 쓰기
         ZigZagBuffer1[i]=0;
         ZigZagBuffer2[i]=0;
        }
      //--- 요일이 정상이면 버퍼를 채웁니다
      else
        {
         //--- 막대 수가 짝수인 경우
         if(i%2==0)
```

© 2000-2025, MetaQuotes Ltd.

```
            {
             //---  첫 번째 버퍼에는 고가, 두 번째 버퍼에는 저가 쓰기
             ZigZagBuffer1[i]=high[i];
             ZigZagBuffer2[i]=low[i];
            }
          //--- 바의 수가 홀수입니다
          else
            {
             //--- 역순으로 막대를 채웁니다
             ZigZagBuffer1[i]=low[i];
             ZigZagBuffer2[i]=high[i];
            }
         }
      }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지그재그 세그먼트의 모양 변경                                    |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- ZigZag 속성에 대한 정보 형성을 위한 문자열
   string comm="";
//--- ZigZag 색상 변경 블록
   int number=MathRand(); // 임의 숫자 가져오기
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);
//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 선 색상 쓰기
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 선 너비 변경 블록
   number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5;   // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
```

© 2000-2025, MetaQuotes Ltd.

```
   int style_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- 계산에 생략된 날짜에 대한 정보 추가
   comm="\r\nNot plotted day - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_FILLING

DRAW_FILLING 스타일은 두 지표 버퍼 값 사이의 색상 영역을 표시합니다. 실제로 이 스타일은 두 개의 선을 그리고 두 개의 지정된 색상 중 하나로 선 사이의 공간을 채웁니다. 채널을 그리는 지표를 만드는 데 사용됩니다. 이 경우에는 아무것도 표시되지 않으므로 어떤 버퍼도 빈 값만 포함할 수 없습니다.

두 가지 채우기 색상을 설정할 수 있습니다:

- 첫 번째 색상은 첫 번째 버퍼의 값이 두 번째 지표 버퍼의 값보다 큰 영역에 사용됩니다;
- 두 번째 색상은 두 번째 버퍼의 값이 첫 번째 지표 버퍼의 값보다 큰 영역에 사용됩니다.

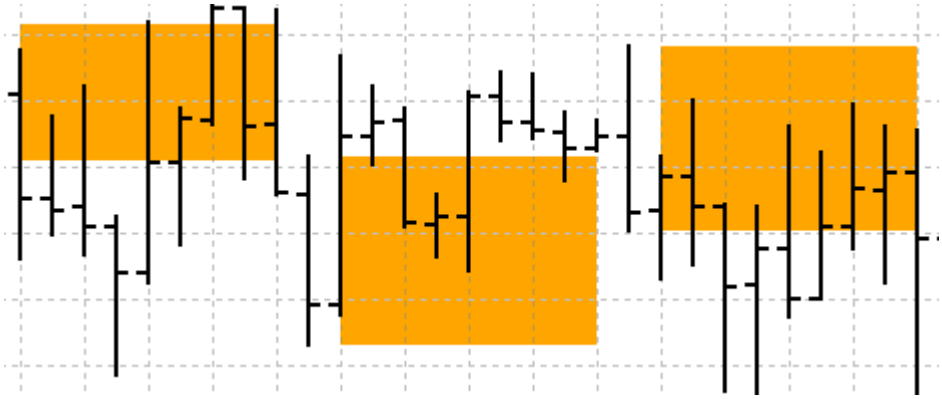
채우기 색상은 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 설정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

지표는 두 지표 버퍼의 값이 0도 아니고 빈 값도 아닌 모든 막대에 대해 계산됩니다. "빈 값"으로 간주할 값을 지정하려면 PLOT_EMPTY_VALUE 속성에 이 값을 설정:

```
   #define INDICATOR_EMPTY_VALUE -1.0
   ...
//--- INDICATOR_EMPTY_VALUE (빈 값)는 다음의 계산에 참여하지 않습니다
   PlotIndexSetDouble (DRAW_FILLING_creation_index,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VA
```

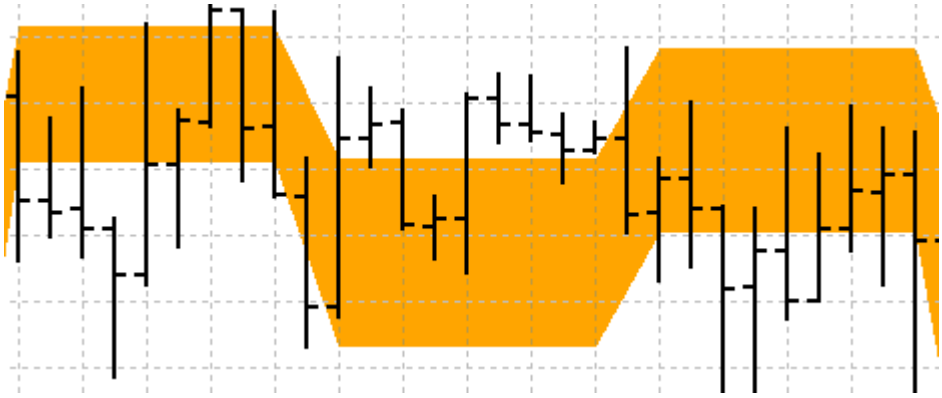
지표 계산에 참여하지 않는 막대에 그리는 작업은 지표 버퍼 값에 따라 달라집니다:

- 두 지표 버퍼의 값이 모두 0인 막대는 지표 그리기에 참여하지 않습니다. 즉, 값이 0인 영역이 채워지지 않습니다.



- 지표 버퍼의 값이 "빈 값"과 같은 막대는 지표 그리기에 참여합니다. 값이 비어 있는 영역은 유의미한 값을 가진 영역을 연결할 수 있도록 채워집니다.

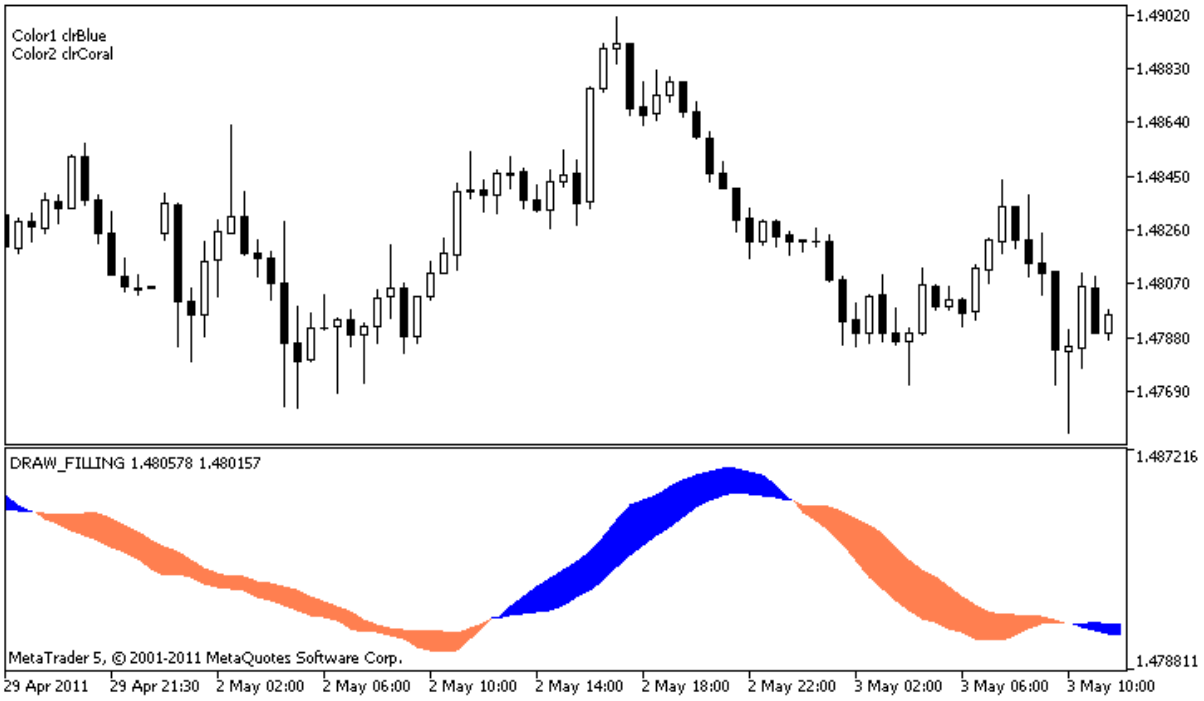
© 2000-2025, MetaQuotes Ltd.



"빈 값"이 0이면 지표 계산에 참여하지 않는 막대도 채워지지 않습니다.

DRAW_FILLING를 플로팅하는 데 필요한 버퍼의 수는 2개입니다.

별도의 창에서 평균 주기가 서로 다른 두 MA 사이에 채널을 그리는 지표의 예. 이동 평균 교차점의 색상 변화는 상승 및 하강 경향의 변화를 시각적으로 보여줍니다. 색상은 **N** 틱마다 임의로 변경됩니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다(지표의 속성 창에 있는 매개변수 탭).



처음에는 DRAW_FILLING의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 속성을 설정한 다음, OnCalculate() 기능에서 새로운 색상이 임의로 설정됩니다.

```
//+------------------------------------------------------------------+
//|                                                DRAW_FILLING.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"

#property description "DRAW_FILLING을 시연하기 위한 지표"
#property description "별도의 창에서 두 개의 MA 사이에 채널을 그립니다"
#property description "채우기 색상은 임의로 변경됩니다"
#property description "모든 N 틱 후"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- Intersection 플롯
#property indicator_label1  "Intersection"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- 매개변수 입력
input int      Fast=13;          // 빠른 MA 기간
input int      Slow=21;          // 느린 MA 기간
input int      shift=1;          // 미래로의 MA 이동(양수)
input int      N=5;              // 변경할 틱의 수
//--- 지표 버퍼
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrC
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
   fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
   slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 충분한 수의 틱이 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표를 처음 계산하거나, 데이터가 변경되어 완전히 다시 계산해야 합니다
   if(prev_calculated==0)
     {
      //--- 모든 지표 값을 적절한 버퍼에 복사
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // 업데이트된 데이터만 채우기
     {
      //--- OnCalculate()의 현재 및 이전 시작 사이의 막대 차이를 가져오기
      int to_copy=rates_total-prev_calculated;
      //--- 차이가 없는 경우에도 0 막대에 하나의 값을 복사합니다
      if(to_copy==0) to_copy=1;
      //--- to_copy 값을 지표 버퍼 맨 끝에 복사
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 채널 채우기 색상 변경                                            |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선의 색상을 변경하기 위한 블록
   int number=MathRand(); // 임의 숫자 가져오기
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);

//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index1=number%size;
//--- 첫 번째 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- 첫 번째 색상 쓰기
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   number=MathRand(); // Get a random number
   int color_index2=number%size;
//--- 두 번째 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- 두 번째 색상 쓰기
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_BARS

DRAW_BARS 스타일은 시가, 고가, 저가 및 종가를 포함하는 네 가지 지표 버퍼 값에 막대를 그립니다. 차트의 별도 하위 창 및 기타 금융상품에 있는 지표를 포함하여 사용자 지정 지표를 막대로 만드는 데 사용됩니다.

막대의 색상은 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 설정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

지표는 지표 버퍼 네 개 **모두** 비어 있지 않은 값이 설정된 막대에만 그려집니다. "빈 값"으로 간주할 값을 지정하려면, PLOT_EMPTY_VALUE 속성에 이 값을 설정:

```
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(index_of_plot_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

항상 지표 버퍼의 값을 명시적으로 입력하고 버퍼에서 빈 값을 건너뛰도록 설정합니다.

DRAW_BARS를 플로팅하는 데 필요한 버퍼의 수는 4개입니다. 플롯을 위한 모든 버퍼는 지정된 순서(시가, 고가, 저가, 종가) 순서대로 하나씩 이동해야 합니다. 이 경우에는 아무것도 표시되지 않으므로 어떤 버퍼도 빈 값만 포함할 수 없습니다.

별도의 창에 선택한 금융상품에 막대를 그리는 지표의 예. 막대 색상은 **N** 틱마다 랜덤하게 변경됩니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다(지표의 속성 창에 있는 매개변수 탭).

plot1의 경우 DRAW_BARS 스타일을 가지고 있으며, 컴파일러 지시문 #property를 사용하여 색상을 설정한 다음, OnCalculate() 기능에서 이전에 준비된 목록에서 색상을 임의로 설정합니다.

```
//+------------------------------------------------------------------+
//|                                                    DRAW_BARS.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_BARS를 시연하기 위한 지표"
#property description "별도의 창에 선택한 심볼의 막대를 그립니다"
#property description "막대의 색상과 너비 및 기호가 임의로 변경됩니다"
#property description "모든 N 틱"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- 막대 플롯
#property indicator_label1  "막대"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      N=5;              // 유형을 변경할 틱의 수
input int      bars=500;         // 표시할 막대의 수
input bool     messages=false;   // "엑스퍼트 어드바이저" 로그에 메시지 표시
//--- 지표 버퍼
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- 심볼 이름
string symbol;
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 막대가 매우 작을 경우 미리 작업을 완료 하십시오
   if(bars<50)
     {
      Comment("더 많은 수의 막대를 지정하십시오! 지표 작동이 종료되었습니다");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 막대가 그려진 심볼의 이름
   symbol=_Symbol;
//--- 심볼 표시 설정
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- 빈 값
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 충분한 수의 틱이 누적된 경우
   if(ticks>=N)
     {
      //--- 마켓 워치 창에서 새 심볼을 선택
      symbol=GetRandomSymbolName();
      //--- 선 속성 변경
      ChangeLineAppearance();

      int tries=0;
      //--- 5 번 시도하여 버퍼에 심볼에서 가격을 표시
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- CopyFromSymbolToBuffers() 기능의 호출 카운터
         tries++;
        }
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 지표 버퍼에 가격 채우기                                  |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- rates[] 배열에서 시가, 고가, 저가, 종가를 복사합니다
   MqlRates rates[];
//--- 시도의 카운터
   int attempts=0;
//--- 복사된 분량
   int copied=0;
//--- 원하는 심볼에 시계열을 35회 입력
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- 충분한 수의 막대를 복사하지 못한 경우
   if(copied!=bars)
     {
      //--- 메시지 문자열 형성
      string comm=StringFormat("심볼 %s의 경우 %d 의 요청된 막대 중 %d 만 수신했습니다",
                               name,
                               copied,
                               bars
                               );
      //--- 기본 차트 창의 코멘트에 메시지 표시
      Comment(comm);
      //--- 메시지 표시
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- 심볼 표시 설정
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- 빈 값으로 버퍼 초기화
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
   ArrayInitialize(BarsBuffer4,0.0);
//--- 가격을 버퍼에 복사
   for(int i=0;i<copied;i++)
     {
      //--- 버퍼의 적절한 인덱스 계산
      int buffer_index=total-copied+i;
      //--- 버퍼에 가격 쓰기
```

© 2000-2025, MetaQuotes Ltd.

```
      BarsBuffer1[buffer_index]=rates[i].open;
      BarsBuffer2[buffer_index]=rates[i].high;
      BarsBuffer3[buffer_index]=rates[i].low;
      BarsBuffer4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| 마켓 워치에서 임의로 심볼을 반환                                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- 마켓 워치 창에 표시되는 심볼의 수
   int symbols=SymbolsTotal(true);
//--- 목록에서 심볼의 포지션 - 0부터 심볼까지의 임의의 숫자
   int number=MathRand()%symbols;
//--- 지정된 포지션에 있는 심볼의 이름을 반환
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| 막대의 모양 변경                                                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 막대 속성에 대한 정보 구성을 위한 문자열
   string comm="";
//--- 막대 색상을 변경하기 위한 블록
   int number=MathRand(); // 임의 숫자 가져오기
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);
//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 선 색상 쓰기
   comm=comm+"\r\n"+(string)colors[color_index];

//--- 막대의 너비 변경을 위한 블록
   number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5;   // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 심볼 이름 쓰기
   comm="\r\n"+symbol+comm;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_CANDLES

DRAW_CANDLES 스타일은 시가, 고가, 저가 및 종가를 포함하는 네 가지 지표 버퍼 값에 캔들스틱을 그립니다. 차트의 별도 하위 창과 기타 금융상품에 있는 항목을 포함하여 일련의 캔들스틱으로서 사용자 지정 지표를 만드는 데 사용됩니다.

캔들스틱의 색상은 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 설정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

지표는 지표 버퍼 네 개 **모두** 비어 있지 않은 값이 설정된 막대에만 그려집니다. "빈 값"으로 간주할 값을 지정하려면, PLOT_EMPTY_VALUE 속성에 이 값을 설정:

```
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(index_of_plot_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

항상 지표 버퍼의 값을 명시적으로 입력하고 버퍼에서 빈 값을 건너뛰도록 설정합니다.

DRAW_CANDLES을 플로팅하는데 필요한 버퍼의 수는 4개입니다. 플롯을 위한 모든 버퍼는 지정된 순서(시가, 고가, 저가, 종가) 순서대로 하나씩 이동해야 합니다. 이 경우에는 아무것도 표시되지 않으므로 어떤 버퍼도 빈 값만 포함할 수 없습니다.

캔들 모양에 영향을 미치는 DRAW_CANDLES 스타일에 대해 최대 세 가지 색상을 설정할 수 있습니다. 색상이 하나만 설정된 경우 차트의 모든 양초에 적용됩니다.

```
//--- 단색이 적용된 같은 캔들
#property indicator_label1  "단일 색상 캔들"
#property indicator_type1   DRAW_CANDLES
//--- 한 가지 색상으로만 지정되므로 모든 캔들이 동일한 색상입니다
#property indicator_color1  clrGreen
```

두개의 쉼표로 구분된 색상을 지정하면 첫 번째 색상을 캔들 윤곽선에 적용하고 두 번째 색상을 본체에 적용합니다.

```
//--- 캔들과 심지의 다른 색상
#property indicator_label1  "두 색상 캔들"
#property indicator_type1   DRAW_CANDLES
//--- 녹색은 심지와 윤곽선에 적용되고, 흰색은 본체에 적용합니다
#property indicator_color1  clrGreen,clrWhite
```

상승 및 하강 캔들이 다르게 표시되도록 쉼표로 구분된 세 가지 색상을 지정. 이 경우, 첫 번째 색상은 캔들의 윤곽선에 적용되고 두 번째와 세 번째 색상은 강세 및 약세 캔들에 적용됩니다.

```
//--- 캔들과 심지의 다른 색상
#property indicator_label1  "단일 색상 캔들"
#property indicator_type1   DRAW_CANDLES
//--- 심지 및 윤곽선은 초록색이고, 강세 캔들 몸통은 흰색이고, 약세 캔들의 몸통은 빨간색입니다
#property indicator_color1  clrGreen,clrWhite,clrRed
```

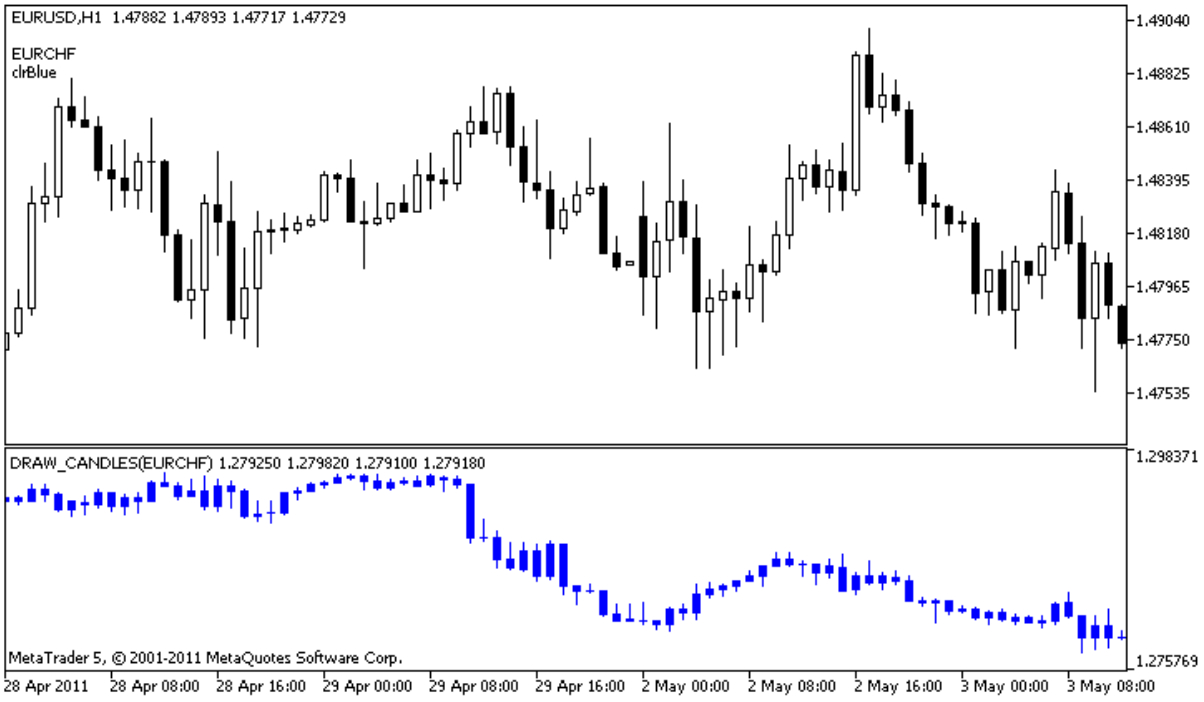
따라서 DRAW_CANDLES 스타일을 사용하면 사용자 지정 캔들 컬러링 옵션을 만들 수 있습니다. 또한 PlotIndexSetInteger 기능(composition_index_DRAW_CANDLES, PLOT_LINE_COLOR, modifier_index, 색상)을 사용하여 지표 작업 중에 모든 색상을 동적으로 변경할 수 있으며, 여기서 modifier_index는 다음과 같은 값을 가질 수 있습니다:

© 2000-2025, MetaQuotes Ltd.

- 0 – 윤곽선과 심지의 색상
- 1 – 강세 캔들 몸통 색상
- 2 – 약세 캔들 몸통 색상

```
//--- 윤곽선과 심지의 색상 지정
PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrBlue);
//--- 강세 몸통 색상 설정
PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrGreen);
//--- 약세 몸통 색상 설정
PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrRed);
```

별도의 창에 선택한 금융상품에 대한 캔들스틱을 그리는 지표의 예. 캔들스틱의 색상은 **N** 틱을 클리할 때마다 임의로 변경됩니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).



plot1의 경우, 컴파일러 지시문 #property를 사용하여 색상을 설정한 다음 OnCalculate() 기능에서 이전에 준비된 목록에서 임의로 색상을 설정합니다.

```
//+------------------------------------------------------------------+
//|                                                 DRAW_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_CANDLES를 시연할 지표."
#property description "별도의 창에 선택한 심볼의 촛대를 그립니다"
#property description " "
#property description "캔들스틱의 색상과 너비 및 심볼이 변경됩니다"
```

```
#property description "모든 N 틱을 임의로"

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- 막대 플롯
#property indicator_label1  "DRAW_CANDLES1"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- 매개변수 입력
input int      N=5;              // 유형을 변경할 틱의 수
input int      bars=500;         // 표시할 막대의 수
input bool     messages=false;   // "엑스퍼트 어드바이저" 로그에 메시지 표시
//--- 지표 버퍼
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- 심볼 이름
string symbol;
//--- 색상을 저장할 배열
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 막대가 매우 작을 경우 미리 작업을 완료 하십시오
   if(bars<50)
     {
      Comment("더 많은 수의 막대를 지정하십시오! 지표 작동이 종료되었습니다");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- 빈 값
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 막대가 그려진 심볼의 이름
   symbol=_Symbol;
//--- 심볼 표시 설정
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 충분한 수의 틱이 누적된 경우
   if(ticks>=N)
     {
      //--- 마켓 워치 창에서 새 심볼을 선택
      symbol=GetRandomSymbolName();
      //--- 양식 변경
      ChangeLineAppearance();
      //--- 마켓 워치 창에서 새 심볼을 선택
      int tries=0;
      //--- 플롯 1의 버퍼를 심볼로 5번 채웁니다
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
         //--- CopyFromSymbolToBuffers() 기능의 호출 카운터
         tries++;
        }
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지정된 캔들스틱을 채웁니다                                       |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
```

© 2000-2025, MetaQuotes Ltd.

```
                              double &buff2[],
                              double &buff3[],
                              double &buff4[]
                              )
  {
//--- rates[] 배열에서 시가, 고가, 저가, 종가를 복사합니다
   MqlRates rates[];
//--- 시도의 카운터
   int attempts=0;
//--- 복사된 분량
   int copied=0;
//--- 원하는 심볼에 시계열을 35회 입력
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- 충분한 수의 막대를 복사하지 못한 경우
   if(copied!=bars)
     {
      //--- 메시지 문자열 형성
      string comm=StringFormat("심볼 %s의 경우 %d 의 요청된 막대 중 %d 만 수신했습니다",
                               name,
                               copied,
                               bars
                               );
      //--- 기본 차트 창의 코멘트에 메시지 표시
      Comment(comm);
      //--- 메시지 표시
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- 심볼 표시 설정
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
     }
//--- 빈 값으로 버퍼 초기화
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- 버퍼에 대한 각 틱 복사 가격
   for(int i=0;i<copied;i++)
     {
      //--- 버퍼의 적절한 인덱스 계산
      int buffer_index=total-copied+i;
      //--- 버퍼에 가격 쓰기
```

© 2000-2025, MetaQuotes Ltd.

```
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| 마켓 워치에서 임의로 심볼을 반환                                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- 마켓 워치 창에 표시되는 심볼의 수
   int symbols=SymbolsTotal(true);
//--- 목록에서 심볼의 포지션 - 0부터 심볼까지의 임의의 숫자
   int number=MathRand()%symbols;
//--- 지정된 포지션에 있는 심볼의 이름을 반환
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| 막대의 모양 변경                                                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 막대 속성에 대한 정보 구성을 위한 문자열
   string comm="";
//--- 막대 색상을 변경하기 위한 블록
   int number=MathRand(); // 임의 숫자 가져오기
//--- 제수는 colors[] 배열의 크기와 같습니다
   int size=ArraySize(colors);
//--- 정수 나눗셈의 나머지의 새로운 색상을 선택할 인덱스를 가져오기
   int color_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- 색상 쓰기
   comm=comm+"\r\n"+(string)colors[color_index];
//--- 심볼 이름 쓰기
   comm="\r\n"+symbol+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_LINE

DRAW_COLOR_LINE 값은 DRAW_LINE 스타일의 색상 변형이며, 지표 버퍼 값을 사용하여 선을 그립니다. 그러나 이 스타일도 제목에 **COLOR**라는 단어가 있는 모든 색상 스타일과 마찬가지로 특별히 설정된 색상 배열의 색상 인덱스(숫자)를 저장하는 추가 특수 표시 버퍼가 있습니다. 따라서 이 막대에 선을 그릴 색상 인덱스를 지정하여 각 선 세그먼트의 색상을 정의할 수 있습니다.

선의 너비, 스타일 및 색상은 컴파일러 지시문과 PlotIndexSetInteger() 기능을 사용하여 동적으로 설정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

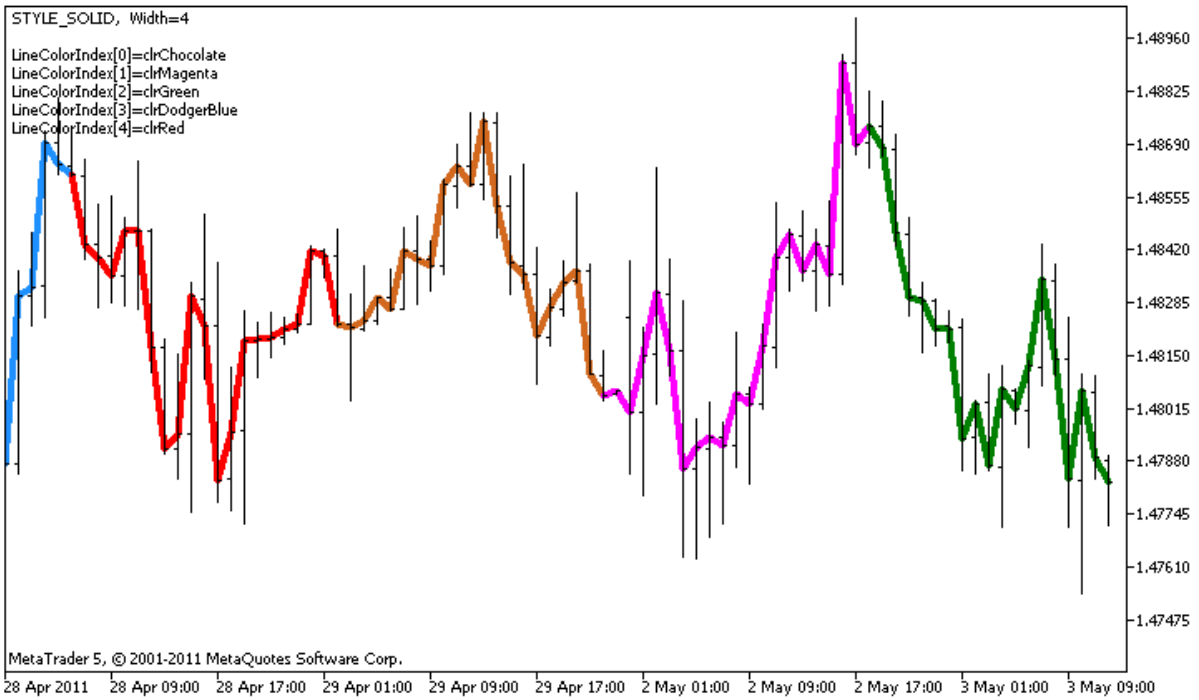
DRAW_COLOR_LINE을 플로팅하는 데 필요한 버퍼의 수는 2개입니다.

- 선을 그리는 데 사용되는 지표 값을 저장하는 버퍼 1개;
- 각 막대에 선 색상 인덱스를 저장하는 버퍼 1개.

색상은 쉼표로 구분된 컴파일러 지시문 #property indicator_color1로 지정할 수 있습니다. 색상 수는 64개를 초과할 수 없습니다.

```
//--- 각 컬러링 막대를 위한 5가지 색상 정의 (특수 배열에 저장됨)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (최대 64
```

막대의 종가를 사용하여 선을 그리는 지표의 예. N=5 틱마다 선 너비와 스타일이 임의로 변경됩니다..



사용자 정의 기능인 ChangeColors()에서도 선 세그먼트의 색상이 임의로 변경됩니다.

```
//+------------------------------------------------------------------+
//| 선 세그먼트의 색상을 변경                                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
```

© 2000-2025, MetaQuotes Ltd.

```
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
                          cols[i]);             //  새 색상
      //--- 색상 쓰기
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

예에서는 지표의 "색상" 버전의 기능을 보여줍니다. 선 세그먼트의 색상을 변경하려면 색상 인덱스가 포함된 ColorLineColors[] 버퍼에서 값을 변경할 필요가 없습니다. 특수 배열에 새 색상을 설정하기만 하면 됩니다. 따라서 PlotIndexSetInteger() 기능을 사용하여 전체 플로팅에 대해 색상을 한 번 빠르게 변경할 수 있습니다.

처음에는 DRAW_COLOR_LINE의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 속성을 설정한 다음, OnCalculate() 기능에서 이 세 속성을 임의로 설정합니다.

N 및 길이(막대 색상 세그먼트 길이) 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수 에서 설정됩니다(지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_LINE을 시연하기 위한 지표"
#property description "종가에 각각 20개의 색상 조각으로 선을 그립니다"
#property description "선의 너비, 스타일 및 색상이 임의로 변경됩니다"
#property description "모든 N 틱"

#property indicator_chart_window
#property indicator_buffers 2
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_plots   1
//--- ColorLine 플롯
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- 각 컬러링 막대를 위한 5가지 색상 정의 (특수 배열에 저장됨)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (최대 64
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      N=5;           //변경할 틱의 수
input int      Length=20;     // 막대에서 각 색상 부분의 길이
int            line_colors=5; // The number of set colors is 5 - see #property indicat
//--- 플로팅할 버퍼
double         ColorLineBuffer[];
//--- 각 막대에 선 색상을 저장하는 버퍼
double         ColorLineColors[];

//--- 색상을 저장하는 배열에는 7가지 요소가 포함됩니다
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 및 배열 바인딩
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- 모조 난수 생성기 초기화
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 선 섹션의 색상 변경
      ChangeColors(colors,5);
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산을 위한 블록
   for(int i=0;i<rates_total;i++)
     {
      //--- 버퍼에 지표 값 쓰기
      ColorLineBuffer[i]=close[i];
      //--- 이제, 이 막대의 색상 인덱스를 임의로 설정
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- 이 막대의 경우, 선은 인덱스 color_index인 색상을 가집니다
      ColorLineColors[i]=color_index;
     }

//--- 함수의 다음 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 선 세그먼트의 색상을 변경                                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
```

© 2000-2025, MetaQuotes Ltd.

```
                              cols[i]);              //  새 색상
      //--- 색상 쓰기
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 지표에 표시된 선의 모양을 변경       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선 너비 변경 블록
   int number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5; // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+" Width="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   int size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm=EnumToString(styles[style_index])+", "+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_SECTION

DRAW_COLOR_SECTION 스타일은 DRAW_SECTION의 색상 버전이지만 후자와는 달리 다양한 색상의 섹션을 그릴 수 있습니다. DRAW_COLOR_SECTION 스타일은 제목에 **COLOR**라는 단어가 있는 모든 색상 스타일과 마찬가지로 특별히 설정된 색상 배열의 색상 인덱스(숫자)를 저장하는 추가 특수 표시 버퍼를 포함합니다. 따라서, 각 섹션의 색상은 섹션 끝에 해당하는 막대의 색상 인덱스를 지정하여 정의할 수 있습니다.

섹션의 너비, ,색상 및 스타일은 DRAW_SECTION 스타일처럼 지정할 수 있으며, 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

섹션이 비어있지 않은 한 값에서 비어 있지 않은 다른 표시 버퍼 값으로 그려지며, 비어 있는 값은 무시됩니다. "빈 값"으로 간주할 값을 지정하려면 PLOT_EMPTY_VALUE 속성에 이 값을 설정: 예를 들어 001 아닌 값에 대해 일련의 섹션으로 지표를 그려야 하는 경우, 0값을 빈 값으로 설정해야 합니다:

```
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

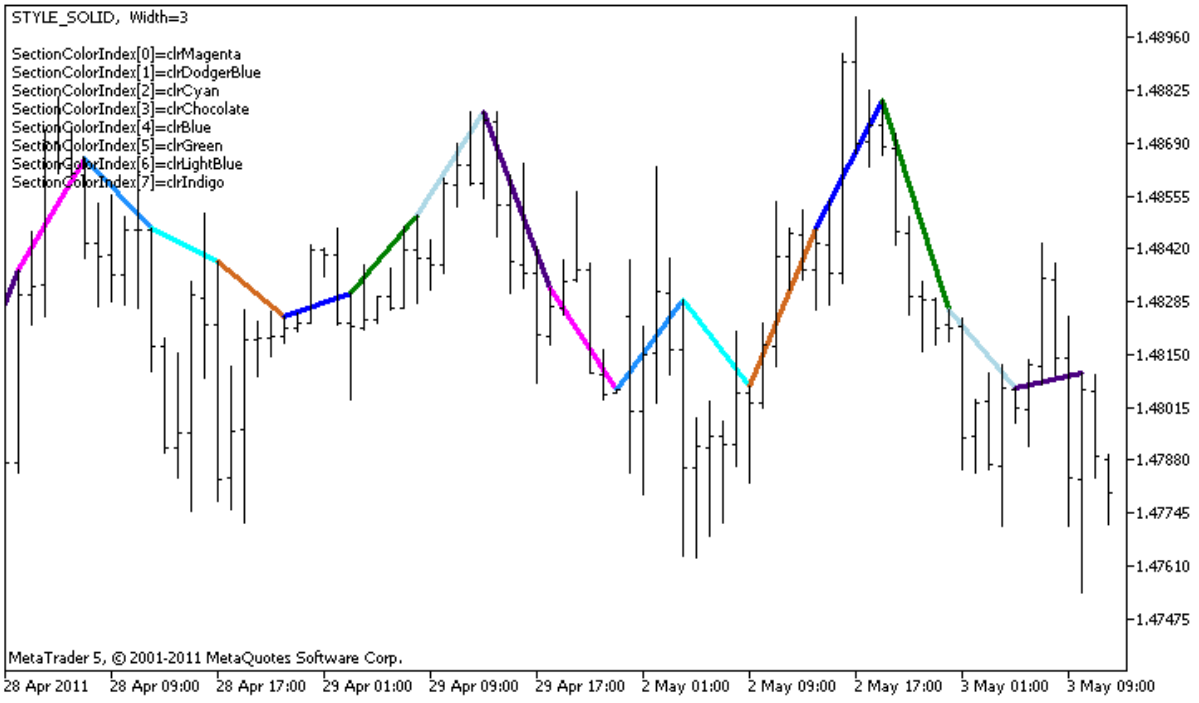
지표 버퍼의 값을 항상 명시적으로 채우고 버퍼의 빈 값을 표시해서는 안 되는 요소로 설정하십시오.

DRAW_COLOR_SECTION을 플로팅하는 데 필요한 버퍼의 수는 2개 입니다.

- 선을 그리는 데 사용되는 지표 값을 저장하는 버퍼 1개;
- 섹션을 그리는 데 사용되는 색상 인덱스를 저장하는 버퍼 1개(비어있지 않은 값만 설정하는 것이 타당함).

색상은 쉼표로 구분된 컴파일러 지시문 #property indicator_color1로 지정할 수 있습니다. 색상 수는 64개를 초과할 수 없습니다.

고가 값을 사용하여 각각 5개 막대 길이의 색상 섹션을 그리는 지표의 예. 섹션의 색상, 너비 및 스타일은 **N** 틱마다 임의로 변경됩니다.



© 2000-2025, MetaQuotes Ltd.

처음에는 DRAW_COLOR_SECTION 8색을 사용하는 **plot1**의 경우 처음 컴파일러 지시문 #property를 사용하여 설정합니다. 그런 다음 OnCalculate() 기능에서 colors[] 색상 배열에서 색상을 임의로 설정합니다.

N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에 설정됩니다(지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_SECTION.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_SECTION을 시연하기 위한 지표"
#property description "지정된 막대 수와 같은 길이의 색상 섹션을 그립니다"
#property description "섹션의 색상, 너비 및 스타일이 임의로 변경됩니다"
#property description "모든 N 틱 후"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- ColorSection 플롯
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- 채색 섹션을 위한 8가지 색상 정의(특수 배열에 저장됨)
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      N=5;                        // 변경할 틱 수
input int      bars_in_section=5;          // 막대에서 섹션의 길이
//--- 섹션의 끝을 계산하는 보조 변수
int            divider;
int            color_sections;
//--- 플로팅할 버퍼
double         ColorSectionBuffer[];
//--- 각 막대에 선 색상을 저장하는 버퍼
double         ColorSectionColors[];
//--- 색상을 저장하는 배열에는 14개의 요소가 포함됩니다
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- 섹션을 채색할 색상의 수
   int color_sections=8;   //   #property indicator_color1에 대한 코멘트 보기
//--- 지표 매개변수 확인
   if(bars_in_section<=0)
     {
      PrintFormat("유효하지 않은 섹션 길이=%d",bars_in_section);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 섹션을 플롯하는 데 사용되는 색상 변경
      ChangeColors(colors,color_sections);
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 지표 값 계산이 시작되는 막대의 수
   int start=0;
//--- 지표가 이전에 계산된 적이 있는 경우 이전 막대에서 시작을 설정하십시오
   if(prev_calculated>0) start=prev_calculated-1;
//--- 다음은 지표 값의 모든 계산값입니다
   for(int i=start;i<rates_total;i++)
     {
      //--- 막대의 번호가 section_length로 나누어진 경우, 이는 섹션의 끝임을 의미합니다
      if(i%bars_in_section==0)
        {
         //--- 섹션 끝을 이 막대의 고가로 설정
         ColorSectionBuffer[i]=high[i];
         //--- 막대 번호를 sceti on_l engt h*number_of_col or s로 나눈 나머지 부분
         int rest=i%divider;
         //색상 = 0 ~ number_of_colors-1까지 번호 가져오기
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //---나눗셈의 나머지가 막대와 같은 경우,
      else
        {
         //--- 아무 일도 없는 경우, 막대 세트 0을 무시
         else ColorSectionBuffer[i]=0;
        }
     }
//--- 함수의 다음 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 선 세그먼트의 색상을 변경                                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
```

© 2000-2025, MetaQuotes Ltd.

```
                              cols[i]);              //  새 색상
      //--- 색상 쓰기
      comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToSt
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 지표에 표시된 선의 모양을 변경       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선 너비 변경 블록
   int number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5; // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+" Width="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   int size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm=EnumToString(styles[style_index])+", "+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM

DRAW_COLOR_HISTOGRAM 스타일은 0부터 지정된 값까지 일련의 색상 열로 히스토그램을 그립니다. 값은 지표 버퍼에서 가져옵니다. 각 열은 미리 정의된 색상 집합에서 고유한 색상을 가질 수 있습니다.

히스토그램의 너비, 색상 및 스타일은 DRAW_HISTOGRAM 스타일과 마찬가지로 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플롯 속성을 동적으로 변경하여 현재 상황에 따라 히스토그램의 모양을 변경할 수 있습니다.

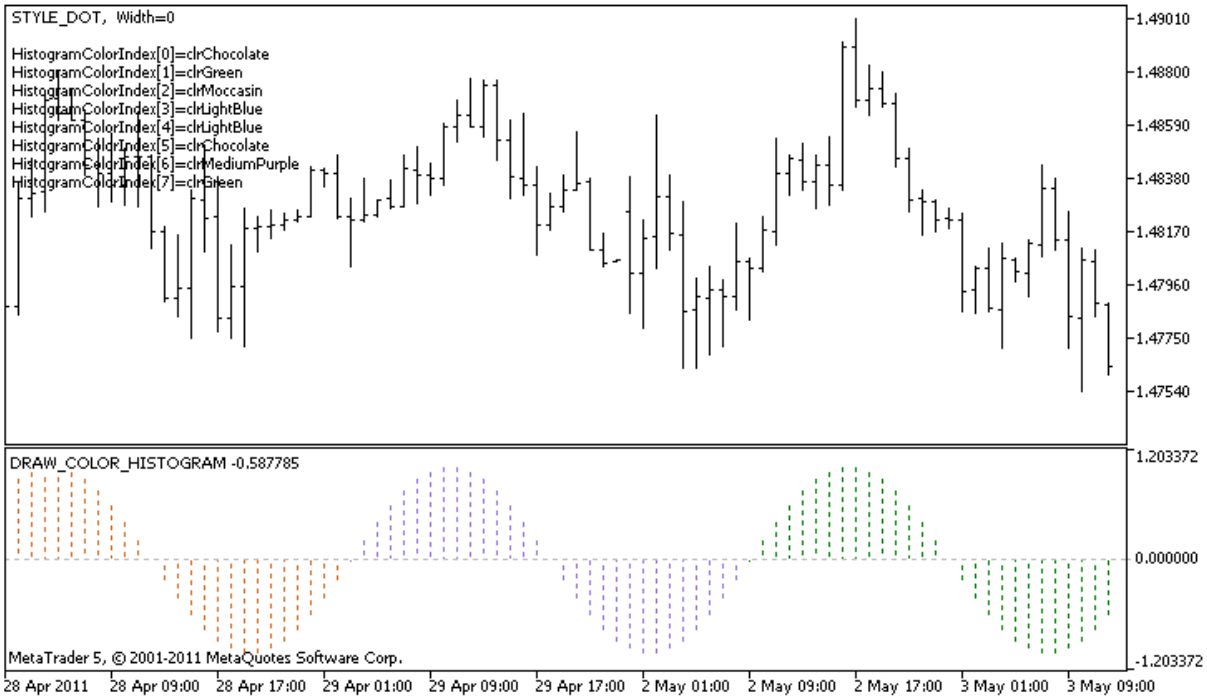
각 막대에 0 레벨의 열이 그려지므로 DRAW_COLOR_HISTOGRAM을 별도의 차트 창에 사용하는 것이 좋습니다. 이러한 유형의 플롯은 대개 오실레이터 유형의 지표를 만드는 데 사용됩니다. 예를 들어, Awesome Oscillator 또는 Market Facilitation Index. 표시할 수 없는 빈 값의 경우 0 값을 지정해야 합니다.

DRAW_COLOR_HISTOGRAM을 플로팅하는 데 필요한 버퍼의 수는 2개입니다.

- 수직 세그먼트의 0이 아닌 값을 각 막대에 저장하기 위한 버퍼 하나, 세그먼트의 두 번째 끝은 항상 지표의 0 라인에 있습니다];
- 섹션을 그리는 데 사용되는 색상 인덱스를 저장하는 버퍼 1개(비어있지 않은 값만 설정하는 것이 타당함).

색상은 쉼표로 구분된 컴파일러 지시문 #property indicator_color1을 사용하여 지정할 수 있습니다. 색상 수는 64개를 초과할 수 없습니다.

지정된 색상의 사인군을 MathSin() 함수 기반으로 그리는 지표의 예. **모든** 히스토그램 열의 색상, 너비 및 스타일이 각 **N** 틱으로 임의 변경됩니다. 막대 매개변수는 사인파의 주기를 지정하며, 이 주기는 지정된 막대의 수 이후이며, 사인파는 사이클을 반복합니다.



DRAW_COLOR_HISTOGRAM 스타일의 **plot1**의 경우 컴파일러 지시문#property를 사용하여 5가지 색상을 설정한 다음 OnCalculate() 기능에서 colors[] 배열에 저장된 14가지 색상 중에서 임의로 색상을 선택합니다.능 N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_HISTOGRAM을 시연하기 위한 지표"
#property description "별도의 창에 사인파를 히스토그램으로 그립니다"
#property description "열의 색상과 너비가 임의로 변경됩니다"
#property description "모든 N 틱 후"
#property description "막대 매개변수는 사인파형을 반복할 막대 수를 설정합니다"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- 매개변수 입력
input int      bars=30;          // 막대의 사인파 주기
input int      N=5;              // 히스토그램을 변경할 틱
//--- Color_Histogram 플롯
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- 채색 섹션을 위한 8가지 색상 정의(특수 배열에 저장됨)
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMe
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 값의 버퍼
double         Color_HistogramBuffer[];
//--- 색상 인덱스의 버퍼
double         Color_HistogramColors[];
//--- 막대 매개변수를 곱산 경우 라디안 단위로 2Pi 각도를 구하는 계수
double         multiplier;
int            color_sections;
//--- 색상을 저장하는 배열에는 14개의 요소가 포함됩니다
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- 사인파를 채색할 색상의 수
   color_sections=8;   //  see A comment to #property indicator_color1
//--- 승수를 계산
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("Set the value of bars=%d greater than 1",bars);
      //--- 지표의 조기 종료
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 히스토그램에 사용되는 색상 변경
      ChangeColors(colors,color_sections);
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산
   int start=0;
//--- 이전 OnCalculate 시작 중에 이미 계산된 경우
   if(prev_calculated>0) start=prev_calculated-1; // 계산의 시작을 단 하나의 막대로 설정
//--- 지표 버퍼에 값을 입력
   for(int i=start;i<rates_total;i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 값
       Color_HistogramBuffer[i]=sin(i*multiplier);
       //--- 색상
       int color_index=i%(bars*color_sections);
       color_index/=bars;
       Color_HistogramColors[i]=color_index;
      }
//--- 함수의 다음 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 선 세그먼트의 색상을 변경                                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
                          cols[i]);             //  새 색상
      //--- 색상 쓰기
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 지표에 표시된 선의 모양을 변경      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선 너비 변경 블록
   int number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
```

© 2000-2025, MetaQuotes Ltd.

```
   int width=number%5; // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+" Width="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   int size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm=EnumToString(styles[style_index])+", "+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_HISTOGRAM2

DRAW_COLOR_HISTOGRAM2 스타일은 두 지표 버퍼 값을 사용하여 지정된 색상(두 지표 버퍼 값을 사용하는 수직 세그먼트)의 히스토그램을 그립니다. 그러나 단색 DRAW_HISTOGRAM2와는 달리 이 스타일에서는 히스토그램의 각 열이 사전 정의된 세트에서 고유한 색상을 가질 수 있습니다. 모든 세그먼트의 값은 지표 버퍼에서 가져옵니다.

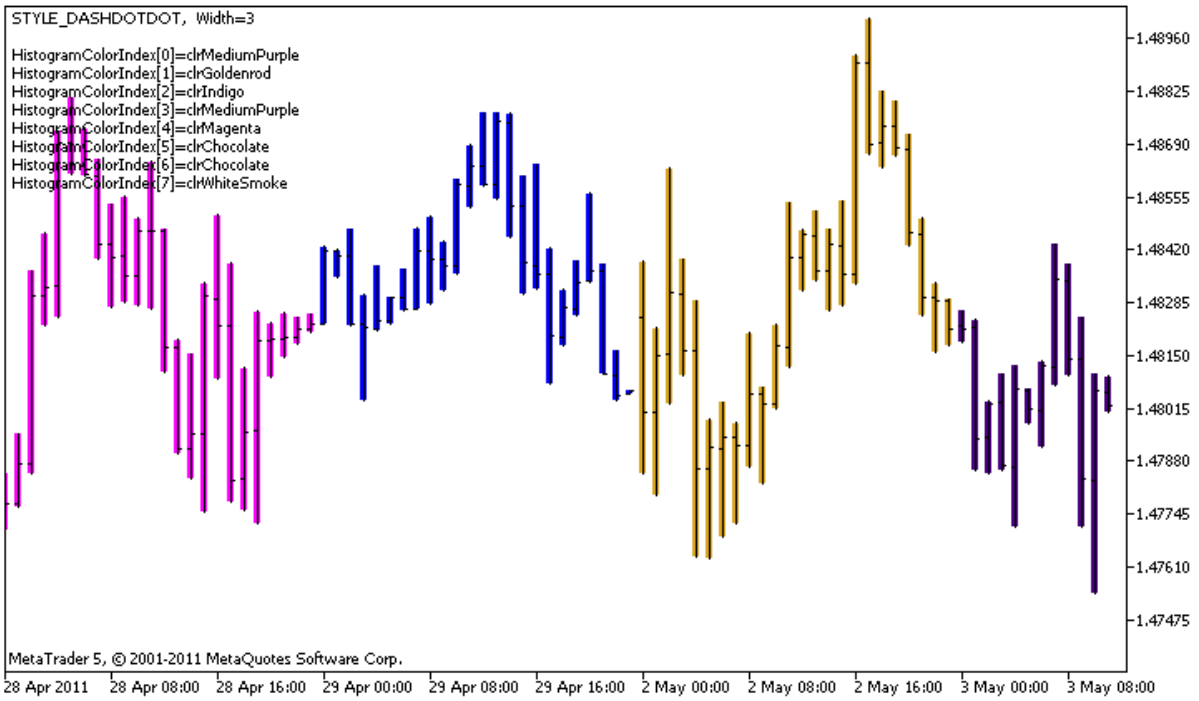
히스토그램의 너비, 스타일 및 색상은 DRAW_HISTOGRAM2 스타일과 마찬가지로 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플롯 속성을 동적으로 변경하여 현재 상황에 따라 히스토그램의 모양을 변경할 수 있습니다.

DRAW_COLOR_HISTOGRAM2 스타일은 차트의 별도의 하위 창과 기본 창에서 사용할 수 있습니다. 빈 값은 아무것도 그려지지 않으므로 지표 버퍼의 모든 값을 명시적으로 설정해야 합니다. 버퍼는 빈 값으로 초기화되지 않습니다.

DRAW_COLOR_HISTOGRAM2를 플로팅하는 데 필요한 버퍼의 수는 3개입니다:

- 각 막대에 수직 세그먼트 상단과 하단을 저장하는 버퍼 2개;
- 세그먼트를 그리는 데 사용되는 색상 인덱스를 저장하는 버퍼 1개(비어 있지 않은 값만 설정하는 것이 타당함).

고가 및 저가 사이에 지정된 색상의 히스토그램을 그리는 지표의 예. 각 요일에 대해 히스토그램 선의 색상이 다릅니다. 히스토그램의 날짜, 너비 및 스타일이 각 N 틱으로 임의 변경됩니다.



DRAW_COLOR_HISTOGRAM2 스타일의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 5가지 색상을 설정한 다음, OnCalculate() 기능에서 colors[] 배열에 저장된 14가지 색상 중에서 임의로 색상을 선택합니다.

N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에 설정됩니다(지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                              DRAW_COLOR_HISTOGRAM2.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_HISTOGRAM2를 시연하기 위한 지표"
#property description "각 막대의 시가 및 종가 사이에 세그먼트를 그립니다"
#property description "색상, 너비 및 스타일이 임의로 변경됩니다"
#property description "모든 N 틱 후"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- ColorHistogram_2 플롯
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- 요일을 기준으로 히스토그램 컬러링을 위한 5가지 색상 정의(특수 배열에 저장됨)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- 매개변수 입력
input int       N=5;              // 히스토그램을 변경할 틱
int             color_sections;
//--- 값 버퍼
double          ColorHistogram_2Buffer1[];
double          ColorHistogram_2Buffer2[];
//--- 색상 인덱스의 버퍼
double          ColorHistogram_2Colors[];
//--- 색상을 저장하는 배열에는 14개의 요소가 포함됩니다
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 빈 값 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- 사인파를 채색할 색상의 수
   color_sections=8;   //  See a comment to #property indicator_color1
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 히스토그램을 그리는 데 사용되는 색상을 변경
      ChangeColors(colors,color_sections);
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산
   int start=0;
//--- 각 막대의 시가별 요일을 가져오기 위해
   MqlDateTime dt;
//--- 이전 OnCalculate 시작 중에 이미 계산된 경우
   if(prev_calculated>0) start=prev_calculated-1; // 계산의 시작을 단 하나의 막대로 설정
//--- 지표 버퍼에 값을 입력
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      //--- 값
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
      //--- 요일에 따라 색상 인덱스를 설정
```

© 2000-2025, MetaQuotes Ltd.

```
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- 함수의 다음 호출에 대해 prev_calculated 값을 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 선 세그먼트의 색상을 변경                                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
                          cols[i]);             //  새 색상
      //--- 색상 쓰기
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 지표에 표시된 선의 모양을 변경      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선 너비 변경 블록
   int number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5; // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+" Width="+IntegerToString(width);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   int size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm=EnumToString(styles[style_index])+", "+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_ARROW

DRAW_COLOR_ARROW 스타일은 지표 버퍼 값을 기반으로 색이 있는 화살표(설정된 Wingdings심볼)를 그립니다. DRAW_ARROW와는 달리, 이 스타일에서는 각 심볼에 대해 indicator_color1 속성으로 지정된 사전 정의된 색상 집합에서 색상을 설정할 수 있습니다.

심볼의 너비와 색상은 DRAW_ARROW 스타일처럼 지정할 수 있으며 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플로팅 특성의 동적 변화를 통해 현재 상황에 따라 지표의 모양을 변경할 수 있습니다.

심볼 코드는 PLOT_ARROW 속성을 사용해서 설정됩니다.

```
//--- PLOT_ARROW에 Wingdings 폰트에서 심볼을 정의
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

PLOT_ARROW=159 의 기본값 (원).

각 화살표는 실제로 높이와 고정점이 있는 기호이며 차트에서 몇 가지 중요한 정보(예: 막대의 종가)를 포함할 수 있습니다. 따라서 차트의 척도에 따라 달라지지 않는 픽셀 단위로 수직 이동을 추가로 지정할 수 있습니다. 화살표는 지정된 픽셀 수만큼 아래로 이동하지만 지표 값은 동일하게 유지됩니다:

```
//--- 화살표의 수직 이동 설정(픽셀)
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

PLOT_ARROW_SHIFT의 음수 값은 위쪽 화살표 이동을 의미하며, 양수 값은 아래쪽 화살표 이동을 의미합니다.

DRAW_COLOR_ARROW 스타일은 차트의 별도 하위 창과 기본 창에서 사용할 수 있습니다. 빈 값은 그려지지 않으며 "데이터 창"에 나타나지 않으니, 지표 버퍼의 모든 값을 명시적으로 설정해야 합니다. 버퍼는 0 값으로 초기화되지 않습니다.

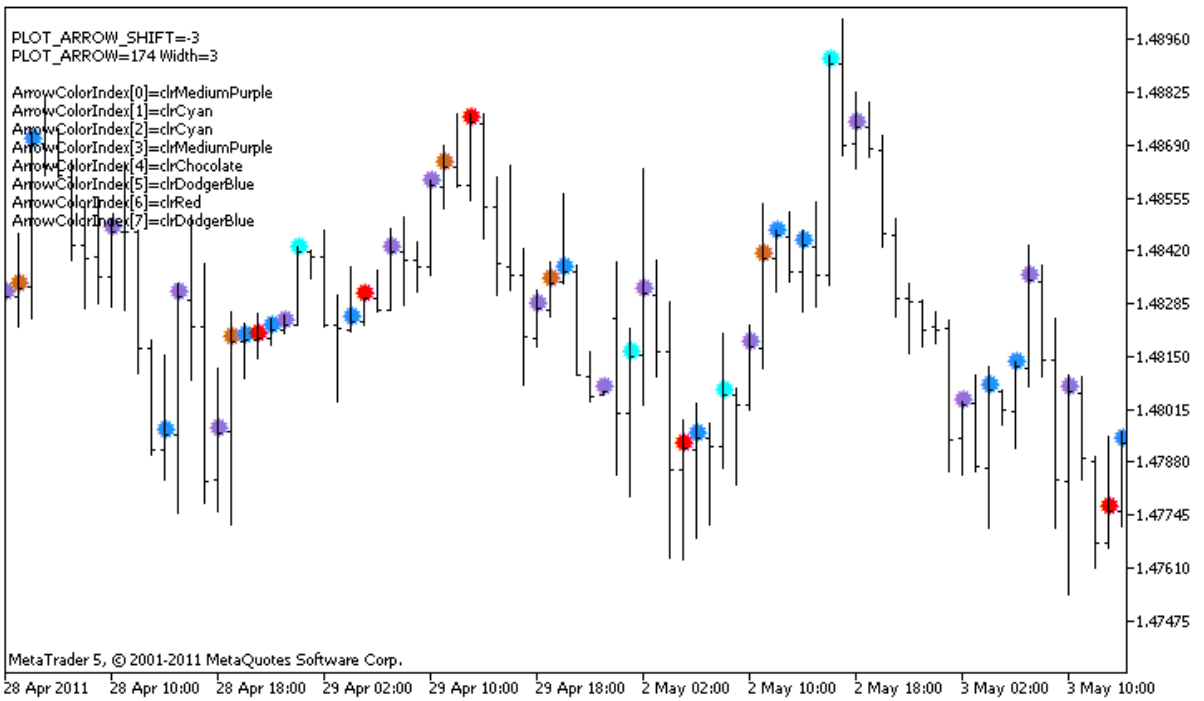
```
//--- 빈 값 설정
   PlotIndexSetDouble(DRAW_COLOR_ARROW_plot_index,PLOT_EMPTY_VALUE,0);
```

DRAW_COLOR_ARROW를 플로팅하는 데 필요한 버퍼의 수는 2개입니다.

- 심볼을 그리는 데 사용되는 가격의 값을 저장하는 버퍼(PLOT_ARROW_SHIFT 속성에 지정된 픽셀 단위 이동);
- 화살표를 그리는 데 사용되는 색상 인덱스를 저장하는 버퍼 (비어 있지 않은 값만 설정하는 것이 타당함).

지표의 예로, 각 막대에 이전 막대의 종가보다 높은 가격으로 화살표를 그립니다. 모든 화살표의 너비, 이동 및 심볼 코드는 N틱마다 임의로 변경됩니다. 심볼의 색상은 심볼이 그려지 막대의 수에 따라 달라집니다.

© 2000-2025, MetaQuotes Ltd.



예에서 **plot1**이 DRAW_COLOR_ARROW 스타일을 사용하는 경우, 컴파일러 지시문 #property를 사용하여 속성, 색상 및 크기를 지정한 다음 OnCalculate() 기능에서 속성을 임의로 설정합니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다 (지표의 속성 창에 있는 매개변수 탭).

컴파일러 지시문 #property를 사용하여 처음에 8가지 색상을 설정하고, OnCalculate() 기능에서 colors[] 배열에 저장된 14가지 색상을 임의로 설정합니다.

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_ARROW.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_ARROW를 시연하기 위한 지표"
#property description "유니코드 문자로 설정된 다양한 색상의 화살표를 차트에 그립니다"
#property description "화살표의 색상, 크기, 이동 및 심볼 코드가 임의로 변경됩니다"
#property description " 모든 N 틱을 임의로"
#property description "코드 매개변수는 기준 값을 설정: code=159 (a circle)"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- ColorArrow 플롯
#property indicator_label1  "ColorArrow"
#property indicator_type1   DRAW_COLOR_ARROW
//--- 히스토그램 컬러링을 위한 8가지 색상 정의 (특수 배열에 저장됨)
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- 매개변수 입력
input int      N=5;          // 변경할 틱의 수
input ushort   code=159;     // DRAW_ARROW에 그릴 심볼 코드
int            color_sections;
//--- 플롯에 대한 지표 버퍼
double         ColorArrowBuffer[];
//--- 색상 인덱스를 저장할 버퍼
double         ColorArrowColors[];
//--- 색상을 저장하는 배열에는 14개의 요소가 포함됩니다
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- PLOT_ARROW에서 도면에 대한 심볼 코드 정의
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- 화살표의 수직 이동 설정(픽셀)
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- 빈 값 0으로 설정
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- 사인파를 채색할 색상의 수
   color_sections=8;   //   #property indicator_color1 에 대한 코멘트 보기
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   static int ticks=0;
//--- 화살표의 색상, 크기, 이동 및 코드를 변경하는 틱 계산
   ticks++;
//--- 틱의 임계 수가 누적된 경우
   if(ticks>=N)
     {
      //--- 화살표 속성 변경
      ChangeLineAppearance();
      //--- 히스토그램을 그리는 데 사용되는 색상을 변경
      ChangeColors(colors,color_sections);
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 지표 값 계산을 위한 블록
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- 계산 루프
   for(int i=1;i<rates_total;i++)
     {
      //--- 현재 종가가 이전 가격보다 높은 경우 화살표를 그립니다
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- 그렇지 않으면 null 값을 지정
      else
         ColorArrowBuffer[i]=0;
      //--- 화살표 색상
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 선 세그먼트의 색상을 변경                                        |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
                          cols[i]);             //  새 색상
      //--- 색상 쓰기
      comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStrin
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 지표에 표시된 선의 모양을 변경       |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 라인 속성에 대한 정보를 구성하기 위한 문자열
   string comm="";
//--- 선 너비 변경 블록
   int number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5; // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+" Width="+IntegerToString(width);

//--- 화살표 코드 변경 블록(PLOT_ARROW)
   number=MathRand();
//--- 화살표의 새 코드(0 ~ 19)를 계산할 수 있는 정수 분할의 나머지를 가져오기
   int code_add=number%20;
//--- code+code_add의 결과로 새 기호 코드를 설정
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- 심볼 코드 PLOT_ARROW 쓰기
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- 화살표의 수직 이동(픽셀)을 변경하기 위한 블록
   number=MathRand();
//--- 이동을 정수 나눗셈의 나머지 값으로 가져오기
   int shift=20-number%41;
//--- 다음에서 새 이동을 설정
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- 이동 PLOT_ARROW_SHIFT 쓰기
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_ZIGZAG

DRAW_COLOR_ZIGZAG 스타일은 두 지표 버퍼 값을 사용하여 서로 다른 색상 세그먼트를 그립니다. 이 스타일은 컬러 버전의 DRAW_ZIGZAG이며 즉, 각 세그먼트에 대해 사전 정의된 색상 집합에서 개별 색상을 지정할 수 있습니다. 세그먼트의 첫 번째 버퍼의 값에서 두 번째 지표 버퍼의 값으로 표시됩니다. 이 경우에는 아무것도 표시되지 않으므로 어떤 버퍼도 빈 값만 포함할 수 없습니다.

선의 너비, 색상 및 스타일을 DRAW_ZIGZAG 스타일과 마찬가지로 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 지정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

섹션은 한 버퍼의 비어 있지 않은 값에서 다른 지표 버퍼의 비어 있지 않은 값으로 그려집니다. "빈 값"으로 간주할 값을 지정하려면 PLOT_EMPTY_VALUE 속성에 이 값을 설정:

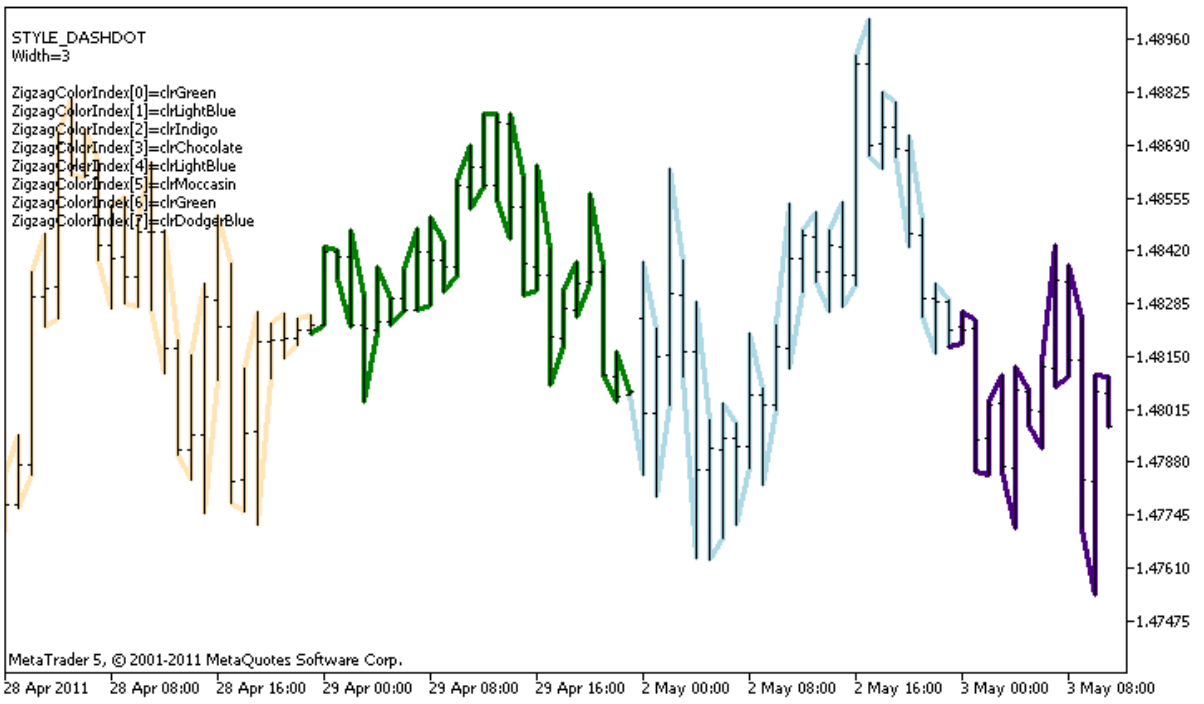
```
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

지표 버퍼를 항상 명시적으로 채우고 버퍼에서 빈 값을 건너뛰도록 설정하십시오.

DRAW_COLOR_ZIGZAG 플로팅에 필요한 버퍼의 수는 3개입니다:

- 지그재그 섹션의 끝 값을 저장하는 버퍼 2개;
- 섹션을 그리는 데 사용되는 색상 인덱스를 저장하는 버퍼 1개(비어있지 않은 값만 설정하는 것이 타당함).

고가와 저가를 기준으로 지그재그를 표시하는 지표의 예. 지그재그 선의 색상, 너비 및 스타일은 **N** 틱마다 임의로 변경됩니다.



DRAW_COLOR_ZIGZAG 스타일의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 8가지 색상을 설정한 다음, OnCalculate() 기능에서 colors[] 배열에 저장된 14가지 색상 중에서 색상을 임의로 선택합니다.

© 2000-2025, MetaQuotes Ltd.

N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에 설정됩니다(지표의 속성 창에 있는 매개변수 탭).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_ZIGZAG를 시연하기 위한 지표"
#property description "색칠된 섹션이 순서로 끊어진 선을 그리며, 색상은 요일의 수에 따라 달라집
#property description "세그먼트의 색상, 너비 및 스타일이 임의로 변경됩니다"
#property description " 모든 N 틱"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   1
//--- Color_Zigzag 플롯
#property indicator_label1  "Color_Zigzag"
#property indicator_type1   DRAW_COLOR_ZIGZAG
//--- 채색 섹션을 위한 8가지 색상 정의(특수 배열에 저장됨)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      N=5;              // 변경할 틱의 수
int            color_sections;
//--- 세그먼트 끝의 값 버퍼
double         Color_ZigzagBuffer1[];
double         Color_ZigzagBuffer2[];
//--- 세그먼트 끝의 색상 인덱스 버퍼
double         Color_ZigzagColors[];
//--- 색상을 저장하는 배열에는 14개의 요소가 포함됩니다
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- 선 스타일을 저장할 배열
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//----지그재그를 채색할 색상의 수
   color_sections=8;   //   #property indicator_color1 속성에 대한 커멘트 보기
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 틱을 계산하여 선의 스타일, 색상 및 너비 변경
   ticks++;
//--- 충분한 수의 틱이 누적된 경우
   if(ticks>=N)
     {
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 섹션을 플롯하는 데 사용되는 색상 변경
      ChangeColors(colors,color_sections);
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }

//--- 시간 구조는 각 막대의 요일을 가져오는 데 필요합니다
   MqlDateTime dt;

//--- 계산 시작 위치
   int start=0;
//--- 지표가 이전 틱에서 계산된 경우, 마지막 틱에서 한 틱으로 계산을 시작합니다
   if(prev_calculated!=0) start=prev_calculated-1;
//--- 계산 루프
   for(int i=start;i<rates_total;i++)
     {
      //--- 구조에 막대 오픈 시간 기록
      TimeToStruct(time[i],dt);
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 막대 숫자가 짝수인 경우
      if(i%2==0)
        {
         //---  첫 번째 버퍼에는 고가, 두 번째 버퍼에는 저가 쓰기
         Color_ZigzagBuffer1[i]=high[i];
         Color_ZigzagBuffer2[i]=low[i];
         //--- 세그먼트의 색상
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
      //--- 막대의 숫자가 홀수
      else
        {
         //--- 역순으로 막대를 채우기
         Color_ZigzagBuffer1[i]=low[i];
         Color_ZigzagBuffer2[i]=high[i];
         //--- 세그먼트의 색상
         Color_ZigzagColors[i]=dt.day_of_year%color_sections;
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지그재그 세그먼트의 색상을 변경                                  |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
                          cols[i]);             //  새 색상
      //--- 색상 쓰기
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 지그재그 세그먼트의 모양 변경                                    |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Color_ZigZag의 속성에 대한 정보 형성을 위한 문자열
   string comm="";
//--- 선 너비 변경 블록
   int number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5;   // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 선의 스타일을 변경하기 위한 블록
   number=MathRand();
//--- 제수는 스타일 배열의 크기와 같습니다
   int size=ArraySize(styles);
//--- 정수 나눗셈의 나머지 항목으로 새 스타일을 선택할 인덱스를 가져오기
   int style_index=number%size;
//--- 색상을 PLOT_LINE_COLOR 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- 선 스타일 쓰기
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_BARS

DRAW_COLOR_BARS 스타일은 시가, 고가, 저가 및 종가를 포함하는 네 가지 지표 버퍼 값에 막대를 그립니다. 이 스타일은 DRAW_BARS의 고급 버전으로, 각 막대에 대해 사전 정의된 색상 집합에서 개별 색상을 지정할 수 있습니다. 차트의 별도 하위 창과 기타 금융 상품에 있는 지표를 포함하여 사용자 지정 지표를 막대로 만드는 데 사용됩니다.

막대의 색상은 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 설정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

지표는 지표 버퍼 네 개 **모두** 비어 있지 않은 값이 설정된 막대에만 그려집니다. "빈 값"으로 간주할 값을 지정하려면, PLOT_EMPTY_VALUE 속성에 이 값을 설정:

```
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

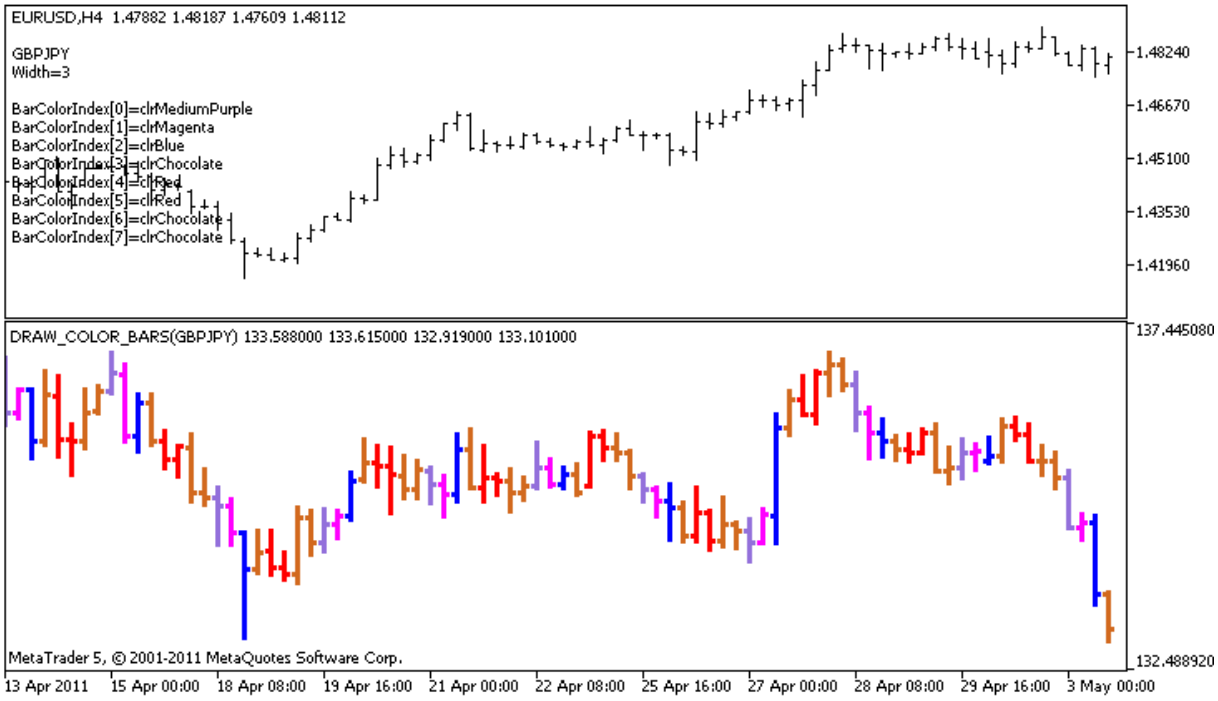
항상 지표 버퍼의 값을 명시적으로 입력하고 버퍼에서 빈 값을 건너뛰도록 설정합니다.

DRAW_COLOR_BARS를 플로팅하는 데 필요한 버퍼 수는 5개 입니다:

- 시가, 고가, 저가 및 종가의 값을 저장하는 4개의 버퍼;
- 색상 인덱스를 저장하는 버퍼 1개는 막대를 그리는 데 사용됩니다(그릴 막대에 대해서만 설정하는 것이 타당함).

플롯을 위한 모든 버퍼는 지정된 순서(시가, 고가, 저가, 종가 및 색상 버퍼)대로 하나씩 이동해야 합니다. 이 경우에는 아무것도 표시되지 않으므로 가격 버퍼에는 빈 값만 포함될 수 없습니다.

별도의 창에 선택한 금융상품에 막대를 그리는 지표의 예. 막대 색상은 **N** 틱마다 랜덤하게 변경됩니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다(지표의 속성 창에 있는 매개변수 탭).



© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_BARS 스타일의 **plot1**의 경우 컴파일러 지시문 #property를 사용하여 8가지 색상을 설정한 다음, OnCalculate() 기능에서 colors[] 배열에 저장된 14가지 색상 중에서 색상을 임의로 선택합니다.

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_BARS.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_BARS를 시연하기 위한 지표"
#property description "별도의 창에 선택한 심볼의 다른 색상 막대를 그립니다"
#property description "막대의 색상과 너비 및 기호가 임의로 변경됩니다"
#property description "모든 N 틱"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- Color Bars 플롯
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- 컬러링 막대에 대한 8가지 색상 정의(특수 배열에 저장됨)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input int      N=5;              // 유형을 변경할 틱의 수
input int      bars=500;         // 표시할 막대의 수
input bool     messages=false;   // "엑스퍼트 어드바이저" 로그에 메시지 표시
//--- 지표 버퍼
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- 심볼 이름
string symbol;
int    bars_colors;
//--- 색상을 저장하는 배열에는 14개의 요소가 포함됩니다
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
//---- Number of colors for coloring bars
   bars_colors=8;   //  see a comment to the #property indicator_color1 property
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=0;
//--- 막대의 스타일, 색상 및 너비를 변경하는 틱의 수
   ticks++;
//--- 충분한 수의 틱이 누적된 경우
   if(ticks>=N)
     {
      //--- 마켓 워치 창에서 새 심볼을 선택
      symbol=GetRandomSymbolName();
      //--- 선 속성 변경
      ChangeLineAppearance();
      //--- 캔들스틱을 그리는 데 사용되는 색상 변경
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- 5 번 시도하여 버퍼에 심볼에서 가격을 표시
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- CopyFromSymbolToBuffers() 기능의 호출 카운터
         tries++;
        }
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지표 버퍼에 가격 채우기                                          |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- rates[] 배열에서 시가, 고가, 저가, 종가를 복사합니다
   MqlRates rates[];
//--- 시도의 카운터
   int attempts=0;
//--- 복사된 분량
   int copied=0;
//--- 원하는 심볼에 시계열을 35회 입력
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
//--- 충분한 수의 막대를 복사하지 못한 경우
   if(copied!=bars)
     {
      //--- 메시지 문자열 형성
      string comm=StringFormat("심볼 %s의 경우 %d 의 요청된 막대 중 %d 만 수신했습니다",
                               name,
                               copied,
                               bars
                               );
      //--- 기본 차트 창의 코멘트에 메시지 표시
      Comment(comm);
      //--- 메시지 표시
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- 심볼 표시 설정
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
//--- 빈 값으로 버퍼 초기화
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- 가격을 버퍼에 복사
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0;i<copied;i++)
     {
      //--- 버퍼의 적절한 인덱스 계산
      int buffer_index=total-copied+i;
      //--- 버퍼에 가격 쓰기
      ColorBarsBuffer1[buffer_index]=rates[i].open;
      ColorBarsBuffer2[buffer_index]=rates[i].high;
      ColorBarsBuffer3[buffer_index]=rates[i].low;
      ColorBarsBuffer4[buffer_index]=rates[i].close;
      //---
      ColorBarsColors[buffer_index]=i%bar_colors;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| 마켓 워치에서 임의로 심볼을 반환                                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- 마켓 워치 창에 표시되는 심볼의 수
   int symbols=SymbolsTotal(true);
//--- 목록에서 심볼의 포지션 - 0부터 심볼까지의 임의의 숫자
   int number=MathRand()%symbols;
//--- 지정된 포지션에 있는 심볼의 이름을 반환
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| 지그재그 세그먼트의 색상을 변경                                  |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
                          cols[i]);             //  새 색상
      //--- 색상 쓰기
```

© 2000-2025, MetaQuotes Ltd.

```
      comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 막대의 모양 변경                                                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 막대 속성에 대한 정보 구성을 위한 문자열
   string comm="";

//--- 막대의 너비 변경을 위한 블록
   int number=MathRand();
//--- 정수 나눗셈의 나머지 너비 가져오기
   int width=number%5;   // 너비는 0 ~ 4로 설정됩니다
//--- 색상을 PLOT_LINE_WIDTH 속성으로 설정
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- 선 너비 쓰기
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- 심볼 이름 쓰기
   comm="\r\n"+symbol+comm;

//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

DRAW_COLOR_CANDLES

DRAW_COLOR_CANDLES 스타일은 DRAW_CANDLES과 마찬가지로 시가, 고가, 저가 및 종가를 포함하는 4개의 지표 버퍼 값을 사용하여 캔들스틱을 그립니다. 또한 특정 세트에서 각 캔들스틱의 색상을 지정할 수 있습니다. 이를 위해 스타일에는 각 막대에 대한 색상 인덱스를 저장하는 특수 색상 버퍼가 있습니다. 또한 차트의 별도 하위 창과 기타 금융상품에 있는 항목을 포함하여 일련의 캔들스틱으로서 사용자 지정 지표를 만드는 데 사용됩니다.

캔들스틱 색상 수는 컴파일러 지시문을 사용하거나 PlotIndexSetInteger() 기능을 사용하여 동적으로 설정할 수 있습니다. 플롯 특성이 동적으로 변경되면 "활기 넣기"가 가능하므로 현재 상황에 따라 지표의 모양이 변경됩니다.

지표는 지표의 가격 버퍼 4개에 대해 비어 있지 않은 값이 설정된 막대에만 표시됩니다. "빈 값"으로 간주할 값을 지정하려면 PLOT_EMPTY_VALUE 속성에 이 값을 설정:

```
//--- 0(빈) 값은 그리기에 포함되지 않습니다
   PlotIndexSetDouble(index_of_plot_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,0);
```

항상 지표 버퍼의 값을 명시적으로 입력하고 버퍼에서 빈 값을 건너뛰도록 설정합니다.

DRAW_COLOR_CANDLS를 플로팅하는 데 필요한 버퍼의 수는 5개 입니다:

- 시가, 고가, 저가 및 종가의 값을 저장하는 4개의 버퍼;
- 색상 인덱스를 저장하는 버퍼 1개, 이 버퍼는 캔들스틱을 그리는 데 사용됩니다(그릴 캔들스틱에 대해서만 설정하는 것이 타당함).

플롯을 위한 모든 버퍼는 지정된 순서(시가, 고가, 저가, 종가 및 색상 버퍼)대로 하나씩 이동해야 합니다. 이 경우에는 아무것도 표시되지 않으므로 가격 버퍼에는 빈 값만 포함될 수 없습니다.

별도의 창에 선택한 금융상품에 대한 캔들스틱을 그리는 지표의 예. 캔들스틱의 색상은 N 틱을 클리할 때마다 임의로 변경됩니다. N 매개변수는 수동 구성 가능성을 위해 지표의 외부 매개변수에서 설정됩니다(지표의 속성 창에 있는 매개변수 탭).

© 2000-2025, MetaQuotes Ltd.

plot1의 경우, 컴파일러 지시문 #property를 사용하여 색상을 설정한 다음 OnCalculate() 기능에서 이전에 준비된 목록에서 임의로 색상을 설정합니다.

```
//+------------------------------------------------------------------+
//|                                            DRAW_COLOR_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_CANDLES를 시연할 지표."
#property description "별도의 창에 선택한 심볼의 촛대를 그립니다"
#property description " "
#property description "캔들스틱의 색상과 너비 및 심볼이 변경됩니다"
#property description "모든 N 틱을 임의로"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- Color Candles 플롯
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- 컬러링 캔들스틱에 8가지 색상을 정의(특수 배열에 저장됨)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- 매개변수 입력
input int      N=5;              // 유형을 변경할 틱의 수
input int      bars=500;         // 표시할 캔들스틱의 수
input bool     messages=false;   // "엑스퍼트 어드바이저" 로그에 메시지 표시
//--- 지표 버퍼
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- 심볼 이름
string symbol;
//--- 색상을 저장하는 배열에는 14개의 요소가 포함됩니다
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 막대가 매우 작을 경우 미리 작업을 완료 하십시오
   if(bars<50)
     {
      Comment("더 많은 수의 막대를 지정하십시오! 지표 작동이 종료되었습니다");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ColorCandlesBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorCandlesBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,ColorCandlesBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,ColorCandlesBuffer4,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- 빈 값
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 막대가 그려진 심볼의 이름
   symbol=_Symbol;
//--- 심볼 표시 설정
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- 캔들스틱을 채색할 색상의 수
   candles_colors=8;      //  참조. #property indicator_color1 속성에 대한 코멘트
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int ticks=INT_MAX-100;
//--- 스타일 및 색상을 변경할 틱의 수
   ticks++;
//--- 충분한 수의 틱이 누적된 경우
   if(ticks>=N)
     {
      //--- 마켓 워치 창에서 새 심볼을 선택
```

© 2000-2025, MetaQuotes Ltd.

```
      symbol=GetRandomSymbolName();
      //--- 양식 변경
      ChangeLineAppearance();
      //--- 캔들스틱을 그리는 데 사용되는 색상 변경
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- 플롯 1의 버퍼를 심볼로 5번 채웁니다
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- CopyFromSymbolToBuffers() 기능의 호출 카운터
         tries++;
        }
      //--- 틱 카운터를 0으로 재설정
      ticks=0;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 지정된 캔들스틱을 채웁니다                                       |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[],
                             double &col_buffer[],
                             int    cndl_colors
                             )
  {
//--- rates[] 배열에서 시가, 고가, 저가, 종가를 복사합니다
   MqlRates rates[];
//--- 시도의 카운터
   int attempts=0;
//--- 복사된 분량
   int copied=0;
//--- 원하는 심볼에 시계열을 35회 입력
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attemp
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 충분한 수의 막대를 복사하지 못한 경우
   if(copied!=bars)
     {
      //--- 메시지 문자열 형성
      string comm=StringFormat("심볼 %s의 경우 %d 의 요청된 막대 중 %d 만 수신했습니다",
                               name,
                               copied,
                               bars
                               );
      //--- 기본 차트 창의 코멘트에 메시지 표시
      Comment(comm);
      //--- 메시지 표시
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- 심볼 표시 설정
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- 빈 값으로 버퍼 초기화
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- 버퍼에 대한 각 틱 복사 가격
   for(int i=0;i<copied;i++)
     {
      //--- 버퍼의 적절한 인덱스 계산
      int buffer_index=total-copied+i;
      //--- 버퍼에 가격 쓰기
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- 캔들스틱 색상 설정
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| 마켓 워치에서 임의로 심볼을 반환                                 |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- 마켓 워치 창에 표시되는 심볼의 수
   int symbols=SymbolsTotal(true);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 목록에서 심볼의 포지션 - 0부터 심볼까지의 임의의 숫자
   int number=MathRand()%symbols;
//--- 지정된 포지션에 있는 심볼의 이름을 반환
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| 캔들스틱 세그먼트의 색상을 변경                                  |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- 색상의 수
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- 각 색상 인덱스에 대해 임의로 새 색상 정의
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- 임의 값 가져오기
      int number=MathRand();
      //--- col[] 배열의 인덱스를 정수 나눗셈의 나머지 항목으로 가져오기
      int i=number%size;
      //--- 각 인덱스의 색상을 PLOT_LINE_COLOR 속성으로 설정
      PlotIndexSetInteger(0,                    //  그래픽 스타일의 수
                          PLOT_LINE_COLOR,      //  속성 식별자
                          plot_color_ind,       //  색의 인덱스, 색상을 쓰는 곳
                          cols[i]);             //  새 색상
      //--- 색상 쓰기
      comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| 캔들스틱의 모양 변경                                             |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- 캔들스틱 속성에 대한 정보 형성을 위한 문자열
   string comm="";
//--- 심볼 이름 쓰기
   comm="\r\n"+symbol+comm;
//--- 설명을 사용하여 차트에 정보 표시
   Comment(comm);
  }
```

© 2000-2025, MetaQuotes Ltd.

지표 속성과 해당 기능 사이의 연결

모든 사용자 지정 지표는 다양한 속성을 가지며, 그 중 일부는 필수적이며 설명 시작 부분에 항상 배치됩니다. 이러한 속성은 다음과 같습니다:

- 지표 플롯을 위한 창의 표시 – indicator_separate_window or indicator_chart_window;
- 지표 버퍼의 수 – indicator_buffers;
- 지표의 플롯 수 – indicator_plots.

또한 전처리기 지시어와 사용자 지정 지표 생성을 위한 기능을 통해 설정할 수 있는 다른 속성도 있습니다. 이러한 특성 및 해당 기능은 다음 표에 설명되어 있습니다.

지표 하위창 특성 지침	IndicatorSet...() 유형의 기능	하위창의 조정된 속성에 대한 설명
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	하위창 높이의 고정 값
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	수직축의 최소값
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	수직축의 최대값
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	N 레벨에 대한 수직축 값
전처리기 지시 없음	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	표시된 레벨의 이름
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	N 레벨 색상
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	N 레벨에 대한 선 너비
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	N 레벨에 대한 선 스타일
특성 플로팅 지침	**PlotIndexSet...() 유형의 기능**	**플롯의 조정된 속성에 대한 설명**
indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	N 숫자 플롯의 짧은 이름. 마우스 커서를 DataWindow과 팝업 도구 설명에 표시됩니다
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	N 플롯에 대한 선 색상

© 2000-2025, MetaQuotes Ltd.

지표 하위창 특성 지침	IndicatorSet...() 유형의 기능	하위창의 조정된 속성에 대한 설명
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	N 플롯에 대한 선 스타일
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	N 플롯에 대한 선 유형
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	Line width for N plot
공통 지표 속성	IndicatorSet...() 유형의 기능	설명
전처리기 지시 없음	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	지표 목록에 표시될 지표의 편리한 짧은 이름을 설정 (**Ctrl+I** 를 눌러 터미널에서 열림).
전처리기 지시 없음	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	지표 값 표시에 필요한 정확도 설정 – 소수 자릿수
전처리기 지시 없음	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	지표 창의 레벨 수를 설정
indicator_applied_price	기능이 없습니다, 속성은 전처리기 지시문으로만 설정할 수 있습니다.	지표 계산에 사용되는 기본 가격 유형. 필요한 경우 첫 번째 유형의 OnCalculate()를 사용하는 경우에만 지정됩니다. 속성 값은 "매개변수" 탭의 지표 속성 대화 상자에서도 설정할 수 있습니다 –"적용 대상".

전처리기 용어의 레벨 및 플롯 수는 1로 시작하는 반면, 함수를 사용할 때의 동일한 속성 수는 0으로 시작하며 즉, 표시된 값이 #property에 대해 표시된 값보다 1 작아야 합니다.

해당하는 기능이 없는 몇 가지 지침이 있습니다:

지시문	설명
indicator_chart_window	지표가 메인 창에 표시됩니다
indicator_separate_window	지표의 별도의 하위창에 표시됩니다
indicator_buffers	필요한 지표 버퍼 수
indicator_plots	지표 내 플롯의 수

© 2000–2025, MetaQuotes Ltd.

SetIndexBuffer

기능은 지정된 지표 버퍼를 double 유형의 1차원 동적 배열과 바인딩합니다.

```
bool  SetIndexBuffer(
   int                   index,          // 버퍼 인덱스
   double                buffer[],       // 배열
   ENUM_INDEXBUFFER_TYPE  data_type      // 저장될 것
   );
```

매개변수

인덱스

[in] 지표 버퍼의 수. 번호는 0으로 시작합니다. 숫자는 #property indicator_buffers에 선언된 값보다 작아야합니다.

buffer[]

[in] 사용자 지정 지표 프로그램에 선언된 배열입니다.

data_type

[in] 지표 배열에 저장된 데이터 유형. 기본적으로 INDICATOR_DATA (계산된 지표의 값)입니다. INDICATOR_COLOR_INDEX의 값도 사용할 수 있으며, 이 경우 이 버퍼는 이전 지표 버퍼의 색상 인덱스를 저장하는 데 사용됩니다. 최대 64가지 색상을 #property indicator_colorN 선에 지정할 수 있습니다. INDICATOR_CALCULATIONS 값은 버퍼가 지표의 중간 계산에 사용되며 그리기용이 아님을 의미합니다.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환합니다.

참고

바인딩 후에는 바인딩된 배열에 대해 시계열 인덱싱을 미리 설치한 경우에도 동적 배열 buffer[]가 공통 배열과 같이 인덱싱됩니다. 지표 배열 요소에 대한 액세스 순서를 변경하려면 ArraySetAsSeries() 기능을 SetIndexBuffer() 기능을 사용하여 배열을 바인딩한 후 사용하십시. 지표 버퍼로 설정된 동적 배열의 크기는 SetIndexBuffer() 함수로 변경할 수 없습니다. 지표 버퍼의 경우 터미널의 실행 하위 시스템에서 크기 변경 작업을 모두 수행합니다.

예:

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer1.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- MA 플롯
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 매개변수 입력
input bool                AsSeries=true;
input int                 period=15;
input ENUM_MA_METHOD      smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE  price=PRICE_CLOSE;
input int                 shift=0;
//--- 지표 버퍼
double                    MABuffer[];
int                       ma_handle;
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 맵핑
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("지표 버퍼는 시계열입니다 = ",ArrayGetAsSeries(MABuffer));
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("SetIndexBuffer()가 시계열인 후에 지표 버퍼 = ",
         ArrayGetAsSeries(MABuffer));

//--- 지표 버퍼 요소 액세스 순서 변경
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- MABuffer 버퍼의 이동 평균 값 복사
```

© 2000-2025, MetaQuotes Ltd.

```
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// AsSeries 값에 따라 다름
                                       // 정말 오래된 값을 수신합니다
                                       // 또는 현재 완료되지 않은 막대에 대하여

//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

추가 참조

사용자 지정 속성, 시계열 및 지표에 대한 액세스

© 2000-2025, MetaQuotes Ltd.

IndicatorSetDouble

기능은 해당 지표 속성의 값을 설정. 지표 속성은 double 유형이어야 합니다. 기능에는 두 가지 종류가 있습니다.

속성 식별자를 지정하여 호출.

```
bool  IndicatorSetDouble(
   int     prop_id,           // 식별자
   double  prop_value         // 설정될 값
   );
```

속성 식별자 및 수정자를 지정하여 호출.

```
bool  IndicatorSetDouble(
   int     prop_id,           // 식별자
   int     prop_modifier,     // 수정자
   double  prop_value         // 설정될 값
   )
```

매개변수

prop_id

[in] 지표 속성의 식별자. 값은 ENUM_CUSTOMIND_PROPERTY_DOUBLE 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정된 속성의 수정자. 레벨 속성만 수정자가 필요합니다. 레벨 번호 지정은 0부터 시작합니다. 즉, 두 번째 레벨에 대한 속성을 설정하려면 1을 지정해야 합니다 (컴파일러 지시문을 사용할 때보다 1이 적음).

prop_value

[in] 속성의 값.

반환 값

실행해 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

참고

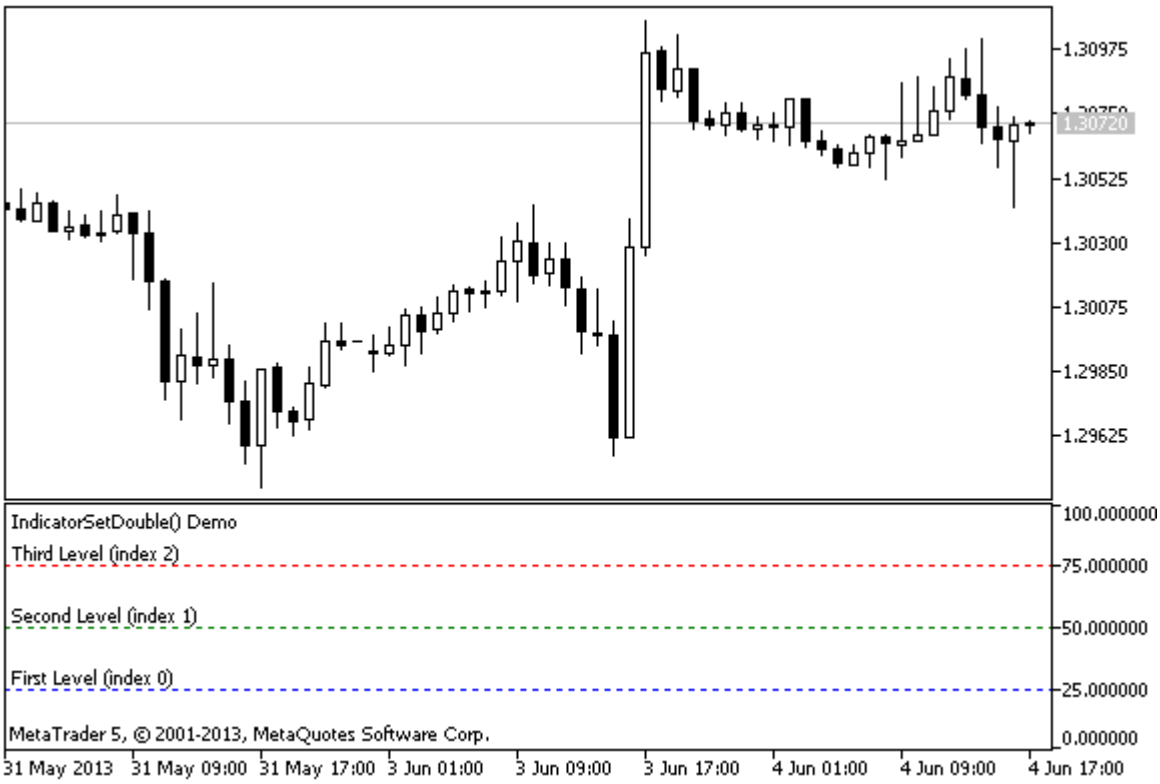
속성(수정자) 번호 지정은 #property 지시문을 사용하는 경우 1부터 시작하고 함수는 0부터 번호 지정. 레벨 번호가 잘못 설정된 경우, 지표 표시가 의도한 것과 다를 수 있습니다.

예를 들어, 별도의 하위 창에서 지표의 첫 번째 레벨 값은 두 가지 방법으로 설정할 수 있습니다:

- 속성 indicator_level**1** 50 -1의 값은 레벨번호를 지정하는데 사용되며,
- IndicatorSetDouble(INDICATOR_LEVELVALUE, **0**, 50) -0 은 첫 번째 레벨을 지정하는 데 사용됩니다.

예: 지표 창의 최대값과 최소값, 수평선이 놓이는 레벨의 값을 뒤집는 지표.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_separate_window
//--- 지표 창의 최대값 및 최소값 설정
#property indicator_minimum  0
#property indicator_maximum  100
//--- 별도의 지표 창에 3개의 수평 레벨을 표시
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- 수평 레벨의 굵기 설정
#property indicator_levelwidth 1
//--- 수평 레벨의 스타일 설정
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 수평 레벨의 설명 설정
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- 지표를 약칭을 설정
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- 각 레벨별로 색을 설정
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
```

© 2000-2025, MetaQuotes Ltd.

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
   static double level1=25,level2=50,level3=75;
   static double max=100,min=0, shift=100;
//--- 틱 계산
   tick_counter++;
//--- 열 번째 틱마다 레벨을 뒤집기
   if(tick_counter%10==0)
     {
      //--- 수준 값에 대한 반전 기호
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- 최대값 및 최소값에 대한 반전 기호
      max-=shift;
      min-=shift;
      //--- 시프트 값을 반전
      shift=-shift;
      //--- 새 레벨 값을 설정
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- 지표 창에서 최대값과 최소값의 새 값을 설정
      Print("Set up max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

추가 참조

© 2000-2025, MetaQuotes Ltd.

예제의 지표 스타일, 지표 속성과 기능 사이의 연결, 그리기 스타일

© 2000-2025, MetaQuotes Ltd.

IndicatorSetInteger

기능은 해당 지표 속성의 값을 설정. 지표 속성은 int 또는 색상 유형이어야 합니다. 기능에는 두 가지 종류가 있습니다.

속성 식별자를 지정하여 호출.

```
bool  IndicatorSetInteger(
   int  prop_id,           // 식별자
   int  prop_value         // 설정될 값
   );
```

속성 식별자 및 수정자를 지정하여 호출.

```
bool  IndicatorSetInteger(
   int  prop_id,           // 식별자
   int  prop_modifier,     // 수정자
   int  prop_value         // 설정될 값
   )
```

매개변수

prop_id

[in] 지표 속성의 식별자. 값은 ENUM_CUSTOMIND_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정된 속성의 수정자. 레벨 속성만 수정자가 필요합니다.

prop_value

[in] 속성의 값.

반환 값

실행해 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

참고

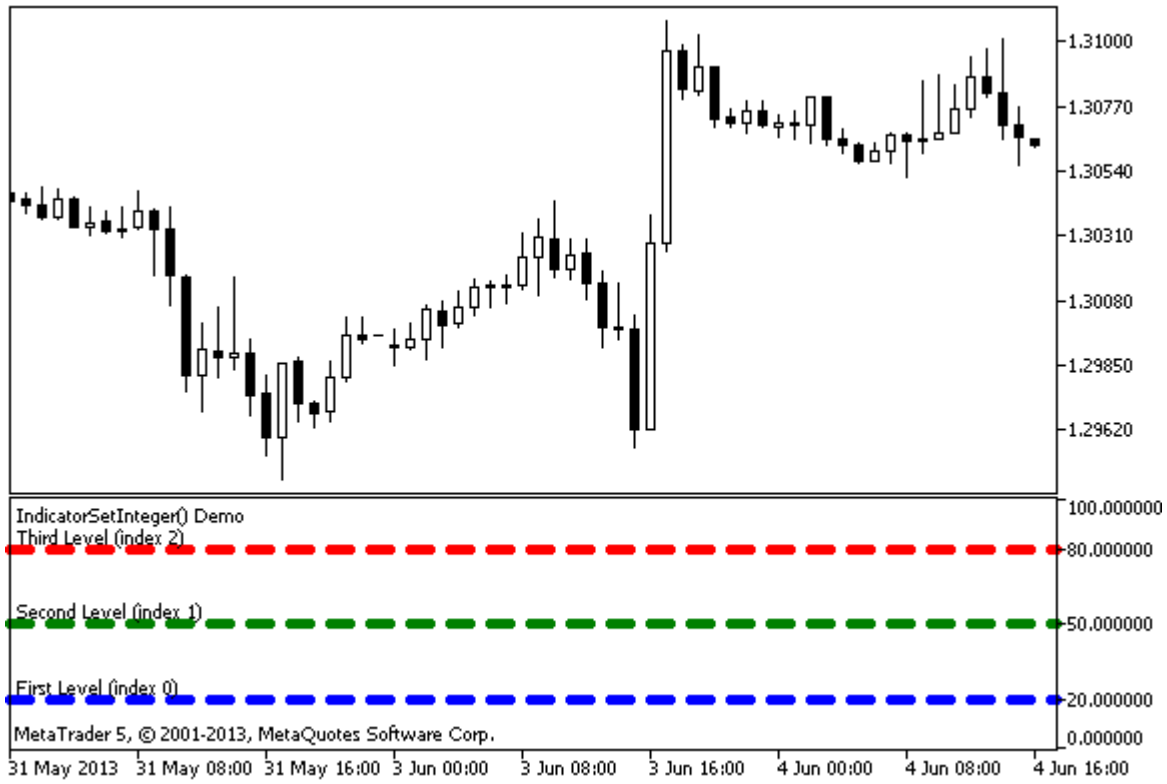
속성(수정자) 번호 지정은 #property 지시문을 사용하는 경우 1부터 시작하고 함수는 0부터 번호 지정. 레벨 번호가 잘못 설정된 경우, 지표 표시가 의도한 것과 다를 수 있습니다.

예를 들어, 첫 번째 수평선의 두께를 설정하려면 0 인덱스를 사용합니다:

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, **0**, 5) – 인덱스 0은 첫 번째 레벨의 두께를 설정하는 데 사용됩니다.

예: 지표 수평선의 색상, 스타일 및 두께를 설정하는 지표.

© 2000-2025, MetaQuotes Ltd.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- 별도의 지표 창에 3개의 수평 레벨을 표시
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//--- 수평 레벨의 굵기 설정
#property indicator_levelwidth 5
//--- 수평선 색상을 설정
#property indicator_levelcolor clrAliceBlue
//--- 수평 레벨의 스타일 설정
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 수평 레벨의 설명 설정
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- 지표를 약칭을 설정
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 사용자 지정 지표 반복 함수                                        |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   static int tick_counter=0;
//--- 틱 계산
   tick_counter++;
//--- 틱 카운터에 따라 수평 레벨의 색상을 계산합니다
   ChangeLevelColor(0,tick_counter,3,6,10); // 세 개의 마지막 매개변수가 색을 스위치
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- 수평 레벨의 스타일을 수정
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- 틱 수를 5로 나눈 나머지 정수 너비로 가져오기
   int width=tick_counter%5;
//--- 모든 수평 레벨에 걸쳐 반복하고 굵기를 설정
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| 별도 지표 창에서 수평선 색상 설정    |
//+------------------------------------------------------------------+
void ChangeLevelColor(int level,      // 수평선의 수
                      int tick_number,// dividend, 나눗셈의 나머지를 얻기 위한 수
                      int f_trigger,  // 색상 전환의 첫 번째 제수
                      int s_trigger,  // 색상 전환의 두 번째 제수
                      int t_trigger)  // 색상 전환의 세 번째 제수
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- colors[] 배열의 색상 인덱스
   int index=-1;
//--- colors[] 배열부터 수평선 칠하기까지 색상 수를 계산
   if(tick_number%f_trigger==0)
      index=0;   // tick_number를 나머지 없이 f_trigger로 나눈 경우
   if(tick_number%s_trigger==0)
      index=1;   // tick_number를 나머지 없이 s_trigger로 나눈 경우
```

© 2000-2025, MetaQuotes Ltd.

```
   if(tick_number%t_trigger==0)
      index=2;   // tick_number를 나머지 없이 t_trigger로 나눈 경우
//--- 색상이 정의되면, 설정
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------+
//| 별도 지표 창에서 수평선 스타일 설정    |
//+------------------------------------------------------------------+
void ChangeLevelStyle(int level,     // 수평선의 수
                      int tick_number// 나머지 나눗셈을 할 수 있는 수
                      )
  {
//--- 스타일을 저장할 배열
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- styles[] 배열의 스타일 인덱스
   int index=-1;
//--- styles[] 배열에서 수평선의 스타일을 설정하기 위한 수를 계산
   if(tick_number%50==0)
      index=5;   // tick_number를 나머지 없이 50으로 나누면 스타일은 STYLE_DASHDOTDOT입니다
   if(tick_number%40==0)
      index=4;   // ... 스타일은 STYLE_DASHDOT입니다
   if(tick_number%30==0)
      index=3;   // ... STYLE_DOT
   if(tick_number%20==0)
      index=2;   // ... STYLE_DASH
   if(tick_number%10==0)
      index=1;   // ... STYLE_SOLID
//--- 스타일이 정의된 경우 설정
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

추가 참조

사용자 지정 지표 속성, 프로그램 속성(#property), 그리기 스타일

© 2000-2025, MetaQuotes Ltd.

IndicatorSetString

기능은 해당 지표 속성의 값을 설정. 지표 속성은 문자열 유형이어야 합니다. 기능에는 두 가지 종류가 있습니다.

속성 식별자를 지정하여 호출.

```
bool  IndicatorSetString(
   int     prop_id,           // 식별자
   string  prop_value         // 설정될 값
   );
```

속성 식별자 및 수정자를 지정하여 호출.

```
bool  IndicatorSetString(
   int     prop_id,           // 식별자
   int     prop_modifier,     // 수정자
   string  prop_value         // 설정될 값
   )
```

매개변수

prop_id

[in] 지표 속성의 식별자. 값은 ENUM_CUSTOMIND_PROPERTY_STRING 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정된 속성의 수정자. 레벨 속성만 수정자가 필요합니다.

prop_value

[in] 속성의 값.

반환 값

실행해 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

참고

속성(수정자) 번호 지정은 #property 지시문을 사용하는 경우 1부터 시작하고 함수는 0부터 번호 지정. 레벨 번호가 잘못 설정된 경우, 지표 표시가 의도한 것과 다를 수 있습니다.

예를 들어, 첫 번째 수평선에 대한 설명을 설정하려면 0 인덱스를 사용합니다:

- IndicatorSetString(INDICATOR_LEVELTEXT, **0**, "First Level") – 인덱스 0은 첫 번째 레벨의 텍스트 설명을 설정하는 데 사용됩니다.

예: 지표는 텍스트 레이블을 지표 수평선에 설정합니다.

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- 별도의 지표 창에 3개의 수평 레벨을 표시
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//--- 수평선 색상을 설정
#property indicator_levelcolor clrRed
//--- 수평 레벨의 스타일 설정
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 수평 레벨의 설명 설정
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- 지표를 약칭을 설정
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---

//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
```

추가 참조

사용자 지정 지표 속성, 프로그램 속성 (#property)

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetDouble

기능은 해당 지표 선의 해당 속성 값을 설정합니다. 지표 속성은 double 유형이어야 합니다.

```
bool  PlotIndexSetDouble(
   int     plot_index,     // 플로팅 스타일 인덱스
   int     prop_id,        // 속성 식별자
   double  prop_value      // 설정될 값
   );
```

매개변수

plot_index

[in] 그래픽 플로팅 인덱스

prop_id

[in] 값은 ENUM_PLOT_PROPERTY_DOUBLE 열거값 중 하나일 수 있습니다.

prop_value

[in] 속성 값.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetInteger

기능은 해당 지표 선의 해당 속성 값을 설정합니다. 지표 속성은 int, char, bool 또는 color 유형이어야 합니다. 이 기능의 변수는 2 가지가 있습니다.

속성의 식별자를 나타내는 호출.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // 플로팅 스타일 인덱스
   int  prop_id,            // 속성 식별자
   int  prop_value          // 설정될 값
   );
```

속성의 식별자 및 수정자를 나타내는 호출.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // 플로팅 스타일 인덱스
   int  prop_id,            // 속성 식별자
   int  prop_modifier,      // 속성 수정자
   int  prop_value          // 설정될 값
   )
```

매개변수

plot_index

[in] 그래픽 플로팅 인덱스

prop_id

[in] 값은 ENUM_PLOT_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정된 속성의 수정자. 색상 인덱스 속성만 수정자가 필요합니다.

prop_value

[in] 속성 값.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환합니다.

예: 삼색선을 그리는 지표. 색배합은 5 틱마다 변경됩니다.

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//---- ColorLine 플롯
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrGreen,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- 지표 버퍼
double          ColorLineBuffer[];
double          ColorBuffer[];
int             MA_handle;
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- MA 핸들 가져오기
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| 색상 인덱스 가져오기                                             |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// 첫 번째 인덱스
   if(j<200) return(1);// 두 번째 인덱스
   return(2); // 세 번째 인덱스
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   static int ticks=0,modified=0;
   int limit;
//--- 첫 번째 계산 또는 막대 수가 변경되었습니다
   if(prev_calculated==0)
     {
      //--- 지표 버퍼 ColorLineBuffer에 MA 값 복사
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// 복사 실패 - 버리기
      //--- 이제 모든 막대의 선 색상을 설정
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- 지표 버퍼 ColorLineBuffer에 MA 값 복사
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// 틱 세기
      if(ticks>=5)//이제 색채를 바꿀 때가 되었습니다
        {
         ticks=0; // 카운터를 재설정
         modified++; // 색상 변경 카운터
         if(modified>=3)modified=0;// 카운터 재설정
         ResetLastError();
         switch(modified)
           {
```

© 2000-2025, MetaQuotes Ltd.

```
            case 0:// 첫 번째 색배합
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("색배합 "+modified);
               break;
            case 1:// 두 번째 색배합
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("색배합 "+modified);
               break;
            default:// 세 번째 색배합
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("색배합 "+modified);
           }
        }
      else
        {
         //--- 시작 포지션 설정
         limit=prev_calculated-1;
         //--- 이제 모든 막대에 대해 선 색상을 설정
         for(int i=limit;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
        }
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

PlotIndexSetString

기능은 해당 지표 선의 해당 속성 값을 설정합니다. 지표 속성은 string 유형이어야 합니다.

```
bool  PlotIndexSetString(
   int     plot_index,     // 플로팅 스타일 인덱스
   int     prop_id,        // 속성 식별자
   string  prop_value      // 설정될 값
   );
```

매개변수

plot_index

[in] 그래픽 플롯 인덱스

prop_id

[in] 값은 ENUM_PLOT_PROPERTY_STRING 열거 값 중 하나일 수 있습니다.

prop_value

[in] 속성 값.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

PlotIndexGetInteger

기능은 해당 지표 선의 해당 속성 값을 설정합니다. 지표 속성은 int, color, bool 또는 char 유형이어야 합니다. 이 기능의 변수는 2 가지가 있습니다.

속성의 식별자를 나타내는 호출.

```
int  PlotIndexGetInteger(
   int  plot_index,        // 플로팅 스타일 인덱스
   int  prop_id,           // 속성 식별자
   );
```

속성의 식별자 및 수정자를 나타내는 호출.

```
int  PlotIndexGetInteger(
   int  plot_index,        // 플로팅 인덱스
   int  prop_id,           // 속성 식별자
   int  prop_modifier      // 속성 수정자
   )
```

매개변수

plot_index

[in] 그래픽 플로팅 인덱스

prop_id

[in] 값은 ENUM_PLOT_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정된 속성의 수정자. 색상 인덱스 속성만 수정자가 필요합니다.

참고

함수는 적절한 지표 선 그리기 설정을 추출하도록 설계 되었습니다. 이 기능은 PlotIndexSetInteger 기능과 함께 작동하여 한 라인의 그리기 속성을 다른 선으로 복사합니다.

예: 요일에 따라 캔들스틱을 채색하는 지표. 각 일별 색상은 프로그래밍 방식으로 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//---- Color Candles 플롯
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 지표 버퍼
double          OpenBuffer[];
double          HighBuffer[];
double          LowBuffer[];
double          CloseBuffer[];
double          ColorCandlesColors[];
color           ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                               clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------+
//| 사용자 지정 지표 초기화 함수                                     |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- 지표 버퍼 맵핑
   SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
   SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- 색상 버퍼에 색상 수 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- 색상 버퍼의 색상을 설정
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- 정확도 설정
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("요일 %i 색상이 있습니다",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------+
//| 사용자 지정 지표 반복 함수                                       |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//---
   int i;
   MqlDateTime t;
//----
   if(prev_calculated==0) i=0;
   else i=prev_calculated-1;
//----
   while(i<rates_total)
     {
      OpenBuffer[i]=open[i];
      HighBuffer[i]=high[i];
      LowBuffer[i]=low[i];
      CloseBuffer[i]=close[i];
      //--- 모든 캔들 색상 설정
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- 다음 호출을 위한 prev_calculated의 반환 값
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

오브젝트 함수

이것은 지정된 차트와 관련된 그래픽 개체로 작업하기 위한 함수 그룹입니다.

차트를 따라 객체를 만들고 이동하기 위한 ObjectCreate() 및 ObjectMove() 작업뿐만 아니라 그래픽 객체의 속성을 정의하는 함수는 실제로 명령을 차트로 전송하는 데 사용됩니다. 이러한 함수가 성공적으로 실행되면 명령은 차트 이벤트의 공통 대기열에 포함됩니다. 차트 이벤트의 대기열을 처리할 때 그래픽 개체의 속성에서 시각적 변화가 구현됩니다.

따라서 이러한 기능을 호출한 후 그래픽 개체를 즉시 시각적 업데이트를 하지 마십시오. 일반적으로 차트에 있는 그래픽 개체는 변경 이벤트 이후에 터미널에 의해 자동으로 업데이트됩니다(새 견적 도착, 차트 창 크기 조정 등). ChartRedraw() 기능을 사용하여 그래픽 개체를 강제로 업데이트합니다.

함수	기능
ObjectCreate	지정된 차트에 지정된 유형의 개체를 생성합니다
ObjectName	지정된 차트(지정된 차트 하위 창)에서 해당 유형의 개체 이름을 반환합니다
ObjectDelete	지정된 차트(지정된 차트 하위 창에서)에서 지정된 이름의 개체를 제거합니다
ObjectsDeleteAll	지정한 차트(지정된 차트 하위 창에서)에서 지정된 유형의 모든 개체를 제거합니다
ObjectFind	이름을 기준으로 지정된 ID를 가진 개체를 검색합니다
ObjectGetTimeByValue	지정된 개체 가격 값에 대한 시간 값을 반환합니다
ObjectGetValueByTime	지정된 시간 동안 개체의 가격 값을 반환합니다
ObjectMove	지정된 개체 고정점의 좌표를 변경합니다
ObjectsTotal	지정된 차트(지정된 차트 하위 창)에서 지정된 유형의 개체 수를 반환합니다
ObjectGetDouble	해당 개체 속성의 double 값을 반환합니다
ObjectGetInteger	해당 개체 속성의 정수 값을 반환합니다
ObjectGetString	해당 개체 속성의 문자열 값을 반환합니다.
ObjectSetDouble	해당 개체 속성의 값을 설정합니다
ObjectSetInteger	해당 개체 속성의 값을 설정합니다
ObjectSetString	해당 개체 속성의 값을 설정합니다
TextSetFont	그리기 방법을 사용하여 텍스트를 표시하는 글꼴을 설정합니다(기본적으로 Arial 20 사용)
TextOut	그래픽 리소스를 생성하도록 설계된 사용자 지정 배열(버퍼)로 텍스트를 전송합니다***
TextGetSize	현재 글꼴 설정에서 문자열의 너비 및 높이를 반환합니다***

© 2000-2025, MetaQuotes Ltd.

모든 그래픽 개체에는 하위 창을 포함하여 하나의 차트내에서 고유한 이름이 있어야 합니다. 그래픽 개체 이름을 변경하면 이전 이름을 가진 개체 삭제 이벤트와 새 이름을 가진 개체 생성 이벤트의 두 가지 이벤트가 생성됩니다.

객체가 생성되거나 객체 속성이 수정된 후에는 ChartRedraw() 함수를 호출하는 것이 좋습니다. 이 함수는 클라이언트 터미널에서 차트(및 그 안에 보이는 모든 객체)를 강제로 그리도록 명령합니다.

© 2000-2025, MetaQuotes Ltd.

ObjectCreate

함수는 지정된 차트 하위 창에 지정된 이름, 유형 및 초기 좌표를 가진 객체를 생성합니다. 생성 중에 최대 30개의 좌표를 지정할 수 있습니다.

```
bool  ObjectCreate(
   long          chart_id,      // 차트 지표
   string        name,          // 객체명
   ENUM_OBJECT   type,          // 객체 유형
   sub_window    nwin,          // 윈도우 인덱스
   datetime      time1,         // 첫 번째 앵커 포인트의 시간
   double        price1,        // 첫 번째 앵커 포인트의 가격
   ...
   datetime      timeN=0,       // N 번째 앵커 포인트의 시간
   double        priceN=0,      // N 번째 앵커 포인트의 가격
   ...
   datetime      time30=0,      // 30 번째 앵커 포인트의 시간
   double        price30=0      // 30 번째 앵커 포인트의 가격
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명. 이름은 하위 창을 포함하여 차트 내에서 고유해야 합니다.

type

[in] 객체 유형. 값은 ENUM_OBJECT 열거값 중 하나일 수 있습니다.

sub_window

[in] 차트 하위 창의 번호. 0은 주 차트 창을 의미합니다. 지정한 하위 창이 있어야 하며 그렇지 않으면 함수가 false를 반환합니다.

time1

[in] 첫 번째 앵커의 시간 좌표.

price1

[in] 첫 번째 앵커의 가격 좌표.

timeN=0

[in] N 번째 앵커의 시간 좌표.

priceN=0

[in] N 번째 앵커의 가격 좌표.

time30=0

[in] 30 번째 앵커의 시간 좌표

price30=0

[in] 30 번째 앵커의 가격 좌표.

© 2000-2025, MetaQuotes Ltd.

반환값

명령이 지정한 차트의 대기열에 성공적으로 추가되면 함수가 true를 반환하고, 그렇지 않으면 false를 반환합니다. 개체가 이미 생성된 경우 개체의 좌표를 변경하려고 시도합니다.

참고

비동기 호출은 항상 ObjectCreate()에 사용되므로 함수는 명령을 차트 대기열에 추가한 결과만 반환합니다. 이 경우 true는 명령이 성공적으로 대기열에 올랐지만 실행 결과를 알 수 없음을 의미합니다.

명령 실행 결과를 확인하려면 ObjectFind() 함수 또는 ObjectGetXXX와 같은 개체 속성을 요청하는 다른 함수를 사용할 수 있습니다. 그러나 이러한 기능은 해당 차트의 대기열 끝에 추가되며 동기 호출로 인해 실행 결과를 기다리기 때문에 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

개체 이름은 63자를 초과할 수 없습니다.

차트 하위 창의 번호 지정(차트에 지표가 있는 하위 창이 있는 경우)은 1로 시작합니다. 차트의 주 차트 창에는 항상 인덱스 0이 있습니다.

향후 사용을 위해 많은 수의 앵커 포인트(최대 30개)가 구현됩니다. 동시에 그래픽 객체에 대해 가능한 앵커 포인트 30개의 한계는 함수를 호출할 때 사용할 수 있는 파라미터 수(64개 이하)에 대한 한계로 결정됩니다.

개체 이름이 변경되면 두 개의 이벤트가 동시에 생성됩니다. 이러한 이벤트는 OnChartEvent() 기능을 통해 Professional Advisor 또는 지표에서 처리할 수 있습니다.:

- 이전 이름을 가진 객체 삭제의 경우
- 새 이름으로 개체를 만드는 이벤트

각 객체 유형을 생성할 때 지정해야 하는 앵커 지점의 수는 다음과 같습니다.

ID	설명	앵커 포인트
OBJ_VLINE	수직선	하나의 앵커 포인트. 사실 시간 좌표만 사용됩니다.
OBJ_HLINE	수평선	하나의 앵커 포인트. 사실 가격 좌표만 사용됩니다.
OBJ_TREND	추세선	두개의 앵커 포인트.
OBJ_TRENDBYANGLE	각도별 추세선	두개의 앵커 포인트.
OBJ_CYCLES	사이클 라인	두개의 앵커 포인트.
OBJ_ARROWED_LINE	애로우 라인	두개의 앵커 포인트.
OBJ_CHANNEL	등거리 채널	세개의 앵커 포인트.
OBJ_STDDEVCHANNEL	표준편차 채널	두개의 앵커 포인트.
OBJ_REGRESSION	선형 회귀 채널	두개의 앵커 포인트.
OBJ_PITCHFORK	Andrews' Pitchfork	세개의 앵커 포인트.
OBJ_GANNLINE	Gann Line	두개의 앵커 포인트.
OBJ_GANNFAN	Gann Fan	두개의 앵커 포인트.

© 2000-2025, MetaQuotes Ltd.

ID	설명	앵커 포인트
OBJ_GANNGRID	Gann Grid	두개의 앵커 포인트.
OBJ_FIBO	피보나치 되돌림	두개의 앵커 포인트.
OBJ_FIBOTIMES	피보나치 시간대	두개의 앵커 포인트.
OBJ_FIBOFAN	피보나치 팬(fan)	두개의 앵커 포인트.
OBJ_FIBOARC	피보나치 호	두개의 앵커 포인트.
OBJ_FIBOCHANNEL	피보나치 채널	세개의 앵커 포인트.
OBJ_EXPANSION	피보나치 확장	세개의 앵커 포인트.
OBJ_ELLIOTWAVE5	엘리엇 모티브 웨이브	5개의 앵커 포인트.
OBJ_ELLIOTWAVE3	엘리엇 수정 파동	세개의 앵커 포인트.
OBJ_RECTANGLE	직사각형	두개의 앵커 포인트.
OBJ_TRIANGLE	삼각형	세개의 앵커 포인트.
OBJ_ELLIPSE	타원	세개의 앵커 포인트.
OBJ_ARROW_THUMB_UP	엄지척	하나의 앵커 포인트.
OBJ_ARROW_THUMB_DOWN	Thumbs Down	하나의 앵커 포인트.
OBJ_ARROW_UP	Arrow Up	하나의 앵커 포인트.
OBJ_ARROW_DOWN	Arrow Down	하나의 앵커 포인트.
OBJ_ARROW_STOP	Stop Sign	하나의 앵커 포인트.
OBJ_ARROW_CHECK	Check Sign	하나의 앵커 포인트.
OBJ_ARROW_LEFT_PRICE	Left Price Label	하나의 앵커 포인트.
OBJ_ARROW_RIGHT_PRICE	Right Price Label	하나의 앵커 포인트.
OBJ_ARROW_BUY	Buy Sign	하나의 앵커 포인트.
OBJ_ARROW_SELL	Sell Sign	하나의 앵커 포인트.
OBJ_ARROW	Arrow	하나의 앵커 포인트.
OBJ_TEXT	Text	하나의 앵커 포인트.
OBJ_LABEL	Label	위치는 OBJPROP_XDISTANCE 및 OBJPROP_YDISTANCE 속성을 사용하여 설정합니다.
OBJ_BUTTON	Button	위치는 OBJPROP_XDISTANCE 및 OBJPROP_YDISTANCE 속성을 사용하여 설정합니다.
OBJ_CHART	Chart	위치는 OBJPROP_XDISTANCE 및 OBJPROP_YDISTANCE 속성을 사용하여 설정합니다.

© 2000-2025, MetaQuotes Ltd.

ID	설명	앵커 포인트
OBJ_BITMAP	Bitmap	하나의 앵커 포인트.
OBJ_BITMAP_LABEL	Bitmap Label	위치는 OBJPROP_XDISTANCE 및 OBJPROP_YDISTANCE 속성을 사용하여 설정합니다.
OBJ_EDIT	Edit	위치는 OBJPROP_XDISTANCE 및 OBJPROP_YDISTANCE 속성을 사용하여 설정합니다.
OBJ_EVENT	경제 달력의 이벤트에 해당하는 "이벤트" 개체	하나의 앵커 포인트. 사실 시간 좌표만 사용됩니다.
OBJ_RECTANGLE_LABEL	커스텀 그래픽 인터페이스를 만들고 설계하기 위한 "직사각형 레이블" 개체.	위치는 OBJPROP_XDISTANCE 및 OBJPROP_YDISTANCE 속성을 사용하여 설정합니다.

© 2000–2025, MetaQuotes Ltd.

ObjectName

함수는 지정된 차트의 지정된 하위 창에서 지정된 유형의 해당 개체 이름을 반환합니다.

```
string  ObjectName(
   long  chart_id,           // 차트 식별자
   int   pos,                // 개체 목록에 있는 숫자
   int   sub_window=-1,      // 윈도우 인덱스
   int   type=-1             // 객체 유형
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

pos

[in] 하위 창의 수와 유형을 기준으로 지정된 필터에 따른 개체의 순서 번호입니다.

sub_window=-1

[in] 차트 하위 창의 번호. 0은 주 차트 창을 의미하며 -1은 주 창을 포함하여 차트의 모든 하위 창을 의미합니다.

type=-1

[in] 객체 유형. 값은 ENUM_OBJECT 열거값 중 하나일 수 있습니다. -1은 모든 유형을 의미합니다.

반환값

성공할 경우 객체 이름이 반환됩니다.

참고

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

개체 이름이 변경되면 두 개의 이벤트가 동시에 생성됩니다. 이러한 이벤트는 OnChartEvent() 기능을 통해 Professional Advisor 또는 지표에서 처리할 수 있습니다.:

- 이전 이름을 가진 객체 삭제의 경우
- 새 이름으로 개체를 만드는 이벤트

© 2000-2025, MetaQuotes Ltd.

ObjectDelete

이 함수는 지정된 차트에서 지정된 이름의 개체를 제거합니다.

```
bool  ObjectDelete(
   long    chart_id,      // chart identifier
   string  name           // object name
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 삭제할 개체의 이름.

반환값

명령이 지정한 차트의 대기열에 성공적으로 추가되면 함수가 true를 반환하고, 그렇지 않으면 false를 반환합니다.

참고

ObjectDelete()에는 항상 비동기 호출이 사용되므로 이 함수는 명령을 차트 대기열에 추가한 결과만 반환합니다. 이 경우 true는 명령이 성공적으로 대기열에 올랐지만 실행 결과를 알 수 없음을 의미합니다.

명령 실행 결과를 확인하려면 ObjectFind() 함수 또는 ObjectGetXXX와 같은 개체 속성을 요청하는 다른 함수를 사용할 수 있습니다. 그러나 이러한 기능은 해당 차트의 대기열 끝에 추가되며 동기 호출로 인해 실행 결과를 기다리기 때문에 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

개체 이름이 변경되면 두 개의 이벤트가 동시에 생성됩니다. 이러한 이벤트는 OnChartEvent() 기능을 통해 Professional Advisor 또는 지표에서 처리할 수 있습니다.:

- 이전 이름을 가진 객체 삭제의 경우
- 새 이름으로 개체를 만드는 이벤트

© 2000-2025, MetaQuotes Ltd.

ObjectsDeleteAll

지정된 유형의 지정된 차트 하위 창에서 모든 개체를 제거합니다.

```
int  ObjectsDeleteAll(
   long  chart_id,           // chart identifier
   int   sub_window=-1,      // 윈도우 인덱스
   int   type=-1             // 객체 유형
   );
```

개체 이름에 접두사를 사용하여 지정된 유형의 모든 개체를 제거합니다.

```
int  ObjectsDeleteAll(
   long           chart_id,   // chart ID
   const string     prefix,   // prefix in object name
   int        sub_window=-1,  // window index
   int      object_type=-1    // object type
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

prefix

[in] 개체명의 접두사. 이름이 이 문자 집합으로 시작하는 모든 개체가 차트에서 제거됩니다. 접두사를 'name' 또는 'name*'으로 지정할 수 있으며 – 두 변수 모두 동일하게 작동합니다. 빈 문자열을 접두사로 지정하면 가능한 모든 이름을 가진 개체가 제거됩니다.

sub_window=-1

[in] 차트 하위 창의 번호. 0은 주 차트 창을 의미하며 -1은 주 창을 포함하여 차트의 모든 하위 창을 의미합니다.

type=-1

[in] 객체 유형. 값은 ENUM_OBJECT 열거값 중 하나일 수 있습니다. -1은 모든 유형을 의미합니다.

반환값

삭제된 개체 수를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

참고

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

© 2000-2025, MetaQuotes Ltd.

ObjectFind

이 함수는 차트에서 지정한 이름의 개체를 지정한 ID로 검색합니다.

```
int  ObjectFind(
   long    chart_id,      // 차트 식별자
   string  name           // object name
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 검색된 개체의 이름입니다.

반환값

이 함수가 성공하면 개체가 발견된 하위 창의 번호(0은 차트의 주 창을 의미합니다)를 반환합니다. 개체를 찾을 수 없으면 함수는 음수를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

참고

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

개체 이름이 변경되면 두 개의 이벤트가 동시에 생성됩니다. 이러한 이벤트는 OnChartEvent() 기능을 통해 Professional Advisor 또는 지표에서 처리할 수 있습니다.:

- 이전 이름을 가진 객체 삭제의 경우
- 새 이름으로 개체를 만드는 이벤트

© 2000-2025, MetaQuotes Ltd.

ObjectGetTimeByValue

이 함수는 지정된 개체의 지정된 가격 값에 대한 시간 값을 반환합니다.

```
datetime  ObjectGetTimeByValue(
   long     chart_id,      // 차트 식별자
   string   name,          // 오브젝트명
   double   value,         // 가격
   int      line_id        // 라인 번호
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

value

[in] 가격 값.

line_id

[in] 라인 식별자.

반환값

지정한 개체의 지정된 가격 값에 대한 시간 값입니다.

참고

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

객체는 하나의 가격 좌표에 여러 개의 값을 가질 수 있으므로 라인 번호를 지정해야 합니다. 이 함수는 다음 개체에만 적용됩니다.

- 추세선 (OBJ_TREND)
- 각도별 추세선 (OBJ_TRENDBYANGLE)
- Gann line (OBJ_GANNLINE)
- 등거리 채널 (OBJ_CHANNEL) - 2 lines
- 선형 회귀 채널 (OBJ_REGRESSION) - 3 lines
- 표준편차 채널 (OBJ_STDDEVCHANNEL) - 3 lines
- 애로우드 라인 (OBJ_ARROWED_LINE)

참고 항목

객체 유형

© 2000-2025, MetaQuotes Ltd.

ObjectGetValueByTime

이 함수는 지정된 객체의 지정된 시간 값에 대한 가격 값을 반환합니다.

```
double  ObjectGetValueByTime(
   long      chart_id,     // 차트 식별자
   string    name,         // 오브젝트명
   datetime  time,         // 시간
   int       line_id       // 라인 번호
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

time

[in] 시간 값.

line_id

[in] Line ID.

반환값

지정된 개체의 지정된 시간 값에 대한 가격 값.

참고

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

객체는 하나의 가격 좌표에 여러 개의 값을 가질 수 있으므로 라인 번호를 지정해야 합니다. 이 함수는 다음 개체에만 적용됩니다.

- 추세선 (OBJ_TREND)
- 각도별 추세선 (OBJ_TRENDBYANGLE)
- Gann line (OBJ_GANNLINE)
- 등거리 채널 (OBJ_CHANNEL) - 2 lines
- 선형 회귀 채널 (OBJ_REGRESSION) - 3 lines
- 표준편차 채널 (OBJ_STDDEVCHANNEL) - 3 lines
- 애로우드 라인 (OBJ_ARROWED_LINE)

참고 항목

객체 유형

© 2000-2025, MetaQuotes Ltd.

ObjectMove

이 함수는 객체의 지정된 앵커 지점의 좌표를 변경합니다.

```
bool  ObjectMove(
   long      chart_id,         // 차트 식별자
   string    name,             // 오브젝트명
   int       point_index,      // 앵커 포인트 넘버
   datetime  time,             // 시간
   double    price             // 가격
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

point_index

[in] 앵커 포인트 인덱스 앵커 지점의 수는 개체 유형에 따라 다릅니다.

time

[in] 선택한 앵커 지점의 시간 좌표입니다.

price

[in] 선택한 앵커 지점의 가격 좌표입니다.

반환값

명령이 지정한 차트의 대기열에 성공적으로 추가되면 함수가 true를 반환하고, 그렇지 않으면 false를 반환합니다.

참고

비동기 호출은 항상 ObjectMove()에 사용되므로 함수는 명령을 차트 대기열에 추가한 결과만 반환합니다. 이 경우 true는 명령이 성공적으로 대기열에 올랐지만 실행 결과를 알 수 없음을 의미합니다.

명령 실행 결과를 확인하려면 ObjectGetXXX와 같은 개체 속성을 요청하는 함수를 사용할 수 있습니다. 그러나 이러한 기능은 해당 차트의 대기열 끝에 추가되며 동기 호출로 인해 실행 결과를 기다리기 때문에 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

© 2000-2025, MetaQuotes Ltd.

ObjectsTotal

이 함수는 지정된 유형의 지정된 차트, 지정된 하위 창에 있는 개체 수를 반환합니다.

```
int  ObjectsTotal(
   long  chart_id,            // 차트 식별자
   int   sub_window=-1,       // 윈도우 인덱스
   int   type=-1              // 객체 유형
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

sub_window=-1

[in] 차트 하위 창의 번호. 0은 주 차트 창을 의미하며 -1은 주 창을 포함하여 차트의 모든 하위 창을 의미합니다.

type=-1

[in] 객체 유형. 값은 ENUM_OBJECT 열거값 중 하나일 수 있습니다. -1은 모든 유형을 의미합니다.

반환값

객체 수.

참고

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

© 2000-2025, MetaQuotes Ltd.

ObjectSetDouble

이 함수는 해당 개체 속성의 값을 설정합니다. 개체 속성은 double 유형이어야 합니다. 이 함수에는 두 가지 변수가 있습니다.

한정자 없이 속성 값 설정

```
bool  ObjectSetDouble(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 개체명
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // 속성
   double                        prop_value         // 값
   );
```

한정자를 나타내는 속성 값 설정

```
bool  ObjectSetDouble(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 개체명
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // 속성
   int                           prop_modifier,     // 한정자
   double                        prop_value         // 값
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

prop_id

[in] 개체 속성의 ID. 이 값은 ENUM_OBJECT_PROPERTY_DOUBLE 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정한 속성의 한정자. 피보나치 툴과 그래픽 객체 앤드류의 피치포크에서 레벨 수를 나타냅니다. 수준 수는 0부터 시작합니다.

prop_value

[in] 속성 값.

반환값

이 함수는 그래픽 개체의 속성을 변경하는 명령이 차트에 성공적으로 전송된 경우에만 true를 반환합니다. 그렇지 않으면 false를 반환하게 됩니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

참고

함수는 비동기 호출을 사용합니다. 이는 함수가 지정된 차트의 대기열에 추가된 명령 실행을 기다리지 않음을 의미합니다. 대신 제어 권한을 즉시 반환합니다.

© 2000-2025, MetaQuotes Ltd.

명령 실행 결과를 확인하려면 지정된 개체 속성을 요청하는 함수를 사용할 수 있습니다. 그러나 이러한 기능은 해당 차트의 대기열 끝에 추가되므로 실행 결과를 기다리기 때문에 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

피보나치 개체를 만들고 개체에 새 수준을 추가하는 예제

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 보조 배열
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//--- 개장 가격 복사 - 최신 바 100개면 충분
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("고가 시리즈 값 복제 실패");
      return;
     }
//--- 폐장 가격 복사 - 최신 바 100개면 충분
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("저가 시리즈 값 복제 실패");
      return;
     }
//--- 마지막 100개의 막대에 대해 오픈 타임 복사
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("시간 가격 값 복제 실패");
      return;
     }
//--- 시계열과 같이 복사된 데이터에 대한 액세스 구성 - 뒤로
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- 피보 객체의 첫 번째 고정점 좌표
   price1=high[70];
   time1=time[70];
//--- 피보 객체의 두 번째 고정점 좌표
   price2=low[50];
   time2=time[50];

//--- Fibo 개체를 생성할 시간
   bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(created) // 개체가 성공적으로 생성된 경우
     {
      //--- 피보 레벨의 색상 설정
      ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
      //--- 그런데 피보 레벨이 얼마나 되나요?
      int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
      Print("Fibo levels before = ",levels);
      //---저널에 출력 => 레벨의 수: value level_description
      for(int i=0;i<levels;i++)
        {
         Print(i,": ",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
               "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
        }
      //--- 단위당 수준 수를 늘림
      bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
      if(!modified) // 수준 수 변경 실패
        {
         Print("피보의 수준 수 변경 실패, 에러 ",GetLastError());
        }
      //--- 단순 통지
      Print("Fibo levels after = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
      //--- 새로 만든 레벨의 값을 설정하다
      bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
      if(added) // 레벨에 대한 값 설정 성공
        {
         Print("피보 레벨을 하나 더 설정했습니다.");
         //--- 또한 레벨 설명을 설정하는 것을 잊지 마십시오
         ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
         ChartRedraw(0);
         //--- Fibo 개체의 수준 수의 실제 값을 가져옵니다.
         levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
         Print("추가 후 피보 레벨 = ",levels);
         //--- 다시 한 번 모든 레벨을 출력합니다 - 단지 확인용
         for(int i=0;i<levels;i++)
           {
            Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                  "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
           }
        }
      else // Fibo 개체의 수준 수를 늘리려고 할 경우 실패함
        {
         Print("하나 이상의 피보 레벨을 설정하는데 실패함. Error ",GetLastError());
        }
     }
  }
```

참고 항목

Object Types, Object Properties

© 2000-2025, MetaQuotes Ltd.

ObjectSetInteger

이 함수는 해당 개체 속성의 값을 설정합니다. 개체속성은 datetime, int, color, bool 또는 char 유형이어야 합니다. 이 함수에는 두 가지 변수가 있습니다.

한정자 없이 속성 값 설정

```
bool  ObjectSetInteger(
   long                              chart_id,        // 차트 식별자
   string                            name,            // 객체명
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,         // 속성
   long                              prop_value       // 값
   );
```

한정자를 나타내는 속성 값 설정

```
bool  ObjectSetInteger(
   long                              chart_id,        // 차트 식별자
   string                            name,            // 객체명
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,         // 속성
   int                               prop_modifier,   // 한정자
   long                              prop_value       // 값
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

prop_id

[in] 개체 속성의 ID. 그 값은 ENUM_OBJECT_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정한 속성의 한정자. 피보나치 툴과 그래픽 객체 앤드류의 피치포크에서 레벨 수를 나타냅니다. 수준 수는 0부터 시작합니다.

prop_value

[in] 속성 값.

반환값

이 함수는 그래픽 개체의 속성을 변경하는 명령이 차트에 성공적으로 전송된 경우에만 true를 반환합니다. 그렇지 않으면 false를 반환하게 됩니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

참고

함수는 비동기 호출을 사용합니다. 이는 함수가 지정된 차트의 대기열에 추가된 명령 실행을 기다리지 않음을 의미합니다. 대신 제어 권한을 즉시 반환합니다.

© 2000-2025, MetaQuotes Ltd.

명령 실행 결과를 확인하려면 지정된 개체 속성을 요청하는 함수를 사용할 수 있습니다. 그러나 이러한 기능은 해당 차트의 대기열 끝에 추가되므로 실행 결과를 기다리기 때문에 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

웹 색상표 작성 예제

```
//+------------------------------------------------------------------+
//|                                                Table of Web Colors|
//|                        Copyright 2011, MetaQuotes Software Corp |
//|                                             https://www.metaquotes.net |
//+------------------------------------------------------------------+
#define X_SIZE 140      // width of an edit object
#define Y_SIZE 33       // height of an edit object
//+------------------------------------------------------------------+
//| 웹 색상 배열                                                     |
//+------------------------------------------------------------------+
color ExtClr[140]=
  {
   clrAliceBlue,clrAntiqueWhite,clrAqua,clrAquamarine,clrAzure,clrBeige,clrBisque,clrB
   clrBlue,clrBlueViolet,clrBrown,clrBurlyWood,clrCadetBlue,clrChartreuse,clrChocolate
   clrCornsilk,clrCrimson,clrCyan,clrDarkBlue,clrDarkCyan,clrDarkGoldenrod,clrDarkGray
   clrDarkMagenta,clrDarkOliveGreen,clrDarkOrange,clrDarkOrchid,clrDarkRed,clrDarkSalm
   clrDarkSlateBlue,clrDarkSlateGray,clrDarkTurquoise,clrDarkViolet,clrDeepPink,clrDee
   clrDodgerBlue,clrFireBrick,clrFloralWhite,clrForestGreen,clrFuchsia,clrGainsboro,cl
   clrGoldenrod,clrGray,clrGreen,clrGreenYellow,clrHoneydew,clrHotPink,clrIndianRed,cl
   clrLavender,clrLavenderBlush,clrLawnGreen,clrLemonChiffon,clrLightBlue,clrLightCora
   clrLightGoldenrod,clrLightGreen,clrLightGray,clrLightPink,clrLightSalmon,clrLightSe
   clrLightSlateGray,clrLightSteelBlue,clrLightYellow,clrLime,clrLimeGreen,clrLinen,cl
   clrMediumAquamarine,clrMediumBlue,clrMediumOrchid,clrMediumPurple,clrMediumSeaGreen
   clrMediumSpringGreen,clrMediumTurquoise,clrMediumVioletRed,clrMidnightBlue,clrMintC
   clrNavajoWhite,clrNavy,clrOldLace,clrOlive,clrOliveDrab,clrOrange,clrOrangeRed,clrC
   clrPaleGreen,clrPaleTurquoise,clrPaleVioletRed,clrPapayaWhip,clrPeachPuff,clrPeru,c
   clrPurple,clrRed,clrRosyBrown,clrRoyalBlue,clrSaddleBrown,clrSalmon,clrSandyBrown,c
   clrSienna,clrSilver,clrSkyBlue,clrSlateBlue,clrSlateGray,clrSnow,clrSpringGreen,cl
   clrThistle,clrTomato,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYell
  };
//+------------------------------------------------------------------+
//| 편집 개체 생성 및 초기화                                         |
//+------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- 새 편집 개체의 이름을 생성합니다
   string name="ColorBox_"+(string)x+"_"+(string)y;
//--- 새 편집 오브젝트를 작성합니다
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("다음을 생성할 수 없음: '",name,"'");
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 좌표, 너비 및 높이 설정
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x*X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y*Y_SIZE);
   ObjectSetInteger(0,name,OBJPROP_XSIZE,X_SIZE);
   ObjectSetInteger(0,name,OBJPROP_YSIZE,Y_SIZE);
//--- 텍스트 색상 설정
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- 배경색 설정
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- 텍스트 설정
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 7x20의 색상 편집 개체 테이블 생성
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

참고 항목

Object Types, Object Properties

© 2000-2025, MetaQuotes Ltd.

ObjectSetString

이 함수는 해당 개체 속성의 값을 설정합니다. 개체 속성은 string 유형이어야 합니다. 이 함수에는 두 가지 변수가 있습니다.

한정자 없이 속성 값 설정

```
bool  ObjectSetString(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 개체명
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // 속성
   string                        prop_value         // 값
   );
```

한정자를 나타내는 속성 값 설정

```
bool  ObjectSetString(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 개체명
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // 속성
   int                           prop_modifier,     // 한정자
   string                        prop_value         // 값
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

prop_id

[in] 개체 속성의 ID. 그 값은 ENUM_OBJECT_PROPERTY_STRING 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정한 속성의 한정자. 피보나치 툴과 그래픽 객체 앤드류의 피치포크에서 레벨 수를 나타냅니다. 수준 수는 0부터 시작합니다.

prop_value

[in] 속성 값.

반환값

이 함수는 그래픽 개체의 속성을 변경하는 명령이 차트에 성공적으로 전송된 경우에만 true를 반환합니다. 그렇지 않으면 false를 반환하게 됩니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

참고

함수는 비동기 호출을 사용합니다. 이는 함수가 지정된 차트의 대기열에 추가된 명령 실행을 기다리지 않음을 의미합니다. 대신 제어 권한을 즉시 반환합니다.

© 2000-2025, MetaQuotes Ltd.

명령 실행 결과를 확인하려면 지정된 개체 속성을 요청하는 함수를 사용할 수 있습니다. 그러나 이러한 기능은 해당 차트의 대기열 끝에 추가되므로 실행 결과를 기다리기 때문에 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

개체 이름이 변경되면 두 개의 이벤트가 동시에 생성됩니다. 이러한 이벤트는 OnChartEvent() 기능을 통해 Professional Advisor 또는 지표에서 처리할 수 있습니다.:

- 이전 이름을 가진 객체 삭제의 경우
- 새 이름으로 개체를 만드는 이벤트

© 2000-2025, MetaQuotes Ltd.

ObjectGetDouble

이 함수는 해당 개체 속성의 값을 반환합니다. 개체 속성은 double 유형이어야 합니다. 이 함수에는 두 가지 변수가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
double  ObjectGetDouble(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 개체명
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // 속성 식별자
   int                           prop_modifier=0    // 속성 한정자(필요한 경우)
   );
```

2. 함수의 성공 여부에 따라 참 또는 거짓을 반환합니다. 성공하면 속성 값이 마지막 매개 변수로 참조로 전달된 수신 변수에 배치됩니다.

```
bool  ObjectGetDouble(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 개체명
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // 속성 식별자
   int                           prop_modifier,     // 속성 한정자
   double&                       double_var         // 여기서 우리는 속성 값을 받아들입
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

prop_id

[in] 개체 속성의 ID. 이 값은 ENUM_OBJECT_PROPERTY_DOUBLE 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정한 속성의 한정자. 첫번째 변수의 경우 기본 한정자 값은 0과 같습니다. 대부분의 속성에는 한정자가 필요하지 않습니다. 피보나치 툴과 그래픽 객체 앤드류의 피치포크에서 레벨 수를 나타냅니다. 수준 수는 0부터 시작합니다.

double_var

[out] 요청한 속성 값을 수신한 double 유형의 변수입니다.

반환값

첫 번째 변수 호출에 대한 double 유형 값입니다.

두 번째 변수의 경우 함수는 true를 반환하며, 이 속성이 유지되고 값이 double_var 변수에 배치된 경우, 그렇지 않으면 false를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

참고

© 2000-2025, MetaQuotes Ltd.

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

© 2000-2025, MetaQuotes Ltd.

ObjectGetInteger

이 함수는 해당 개체 속성의 값을 반환합니다. 개체속성은 datetime, int, color, bool 또는 char 유형이어야 합니다. 이 함수에는 두 가지 변수가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
long  ObjectGetInteger(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 오브젝트명
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // 속성 식별자
   int                           prop_modifier=0    // 속성 한정자(필요한 경우)
   );
```

2. 함수의 성공 여부에 따라 참 또는 거짓을 반환합니다. 성공하면 속성 값이 마지막 매개 변수로 참조로 전달된 수신 변수에 배치됩니다.

```
bool  ObjectGetInteger(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 오브젝트명
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // 속성 식별자
   int                           prop_modifier,     // 속성 식별자
   long&                         long_var           // 여기서 우리는 속성 값을 받아들입
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

prop_id

[in] 개체 속성의 ID. 그 값은 ENUM_OBJECT_PROPERTY_INTEGER 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정한 속성의 한정자. 첫번째 변수의 경우 기본 한정자 값은 0과 같습니다. 대부분의 속성에는 한정자가 필요하지 않습니다. 피보나치 툴과 그래픽 객체 앤드류의 피치포크에서 레벨 수를 나타냅니다. 수준 수는 0부터 시작합니다.

long_var

[out] 요청한 속성의 값을 받는 long 유형의 변수입니다.

반환값

첫 번째 호출 변수에 대한 long 값입니다.

두 번째 변종의 경우 함수는 true를 반환하며, 이 속성이 유지되고 값이 long_var 변수에 배치된 경우, 그렇지 않으면 false를 반환하지 않으면 false를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

참고

© 2000-2025, MetaQuotes Ltd.

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

© 2000-2025, MetaQuotes Ltd.

ObjectGetString

이 함수는 해당 개체 속성의 값을 반환합니다. 개체 속성은 string 유형이어야 합니다. 이 함수에는 두 가지 변수가 있습니다.

1. 속성 값을 즉시 반환합니다.

```
string  ObjectGetString(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 개체명
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // 속성 식별자
   int                           prop_modifier=0    // 속성 한정자(필요한 경우)
   );
```

2. 함수의 성공 여부에 따라 참 또는 거짓을 반환합니다. 성공하면 속성 값이 마지막 매개 변수로 참조로 전달된 수신 변수에 배치됩니다.

```
bool  ObjectGetString(
   long                          chart_id,          // 차트 식별자
   string                        name,              // 개체명
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // 속성 식별자
   int                           prop_modifier,     // 속성 한정자
   string&                       string_var         // 여기서 우리는 속성 값을 받아들이
   );
```

Parameter

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

name

[in] 객체명.

prop_id

[in] 개체 속성의 ID. 그 값은 ENUM_OBJECT_PROPERTY_STRING 열거값 중 하나일 수 있습니다.

prop_modifier

[in] 지정한 속성의 한정자. 첫번째 변수의 경우 기본 한정자 값은 0과 같습니다. 대부분의 속성에는 한정자가 필요하지 않습니다. 피보나치 툴과 그래픽 객체 앤드류의 피치포크에서 레벨 수를 나타냅니다. 수준 수는 0부터 시작합니다.

string_var

[out] 요청한 속성 값을 수신하는 문자열 유형의 변수입니다.

반환값

첫 번째 버전 호출의 string 값.

이 속성이 유지 관리되고 값이 string_var 변수에 배치된 경우, 두 번째 버전의 호출에서는 true를 반환하고, 그렇지 않으면 false를 반환합니다. 에러에 대한 자세한 내용은 GetLastError()를 참조하십시오.

참고

© 2000-2025, MetaQuotes Ltd.

이 함수는 동기식 호출을 사용합니다. 이는 이 함수가 호출하기 전에 이 차트에 대해 대기한 모든 명령의 실행을 대기하므로 이 함수는 시간이 많이 걸릴 수 있습니다. 차트에서 많은 개체로 작업할 때는 이 기능을 고려해야 합니다.

개체 이름이 변경되면 두 개의 이벤트가 동시에 생성됩니다. 이러한 이벤트는 OnChartEvent() 기능을 통해 Professional Advisor 또는 지표에서 처리할 수 있습니다.:

- 이전 이름을 가진 객체 삭제의 경우
- 새 이름으로 개체를 만드는 이벤트

© 2000-2025, MetaQuotes Ltd.

TextSetFont

이 함수는 그리기 방법을 사용하여 텍스트를 표시하는 글꼴을 설정하고 해당 작업의 결과를 반환합니다. 크기가 -120(12pt)인 Arial 글꼴이 기본적으로 사용됩니다.

```
bool  TextSetFont(
   const string  name,           // 디스크의 폰트명 또는 폰트 파일의 경로
   int           size,           // 폰트 사이즈
   uint          flags,          // 플래그 조합
   int           orientation=0   // 텍스트 기울기 각도
   );
```

Parameter

name

[in] 시스템의 폰트명 또는 디스크에 있는 폰트 파일 또는 폰트 경로를 포함하는 리소스의 이름입니다.

size

[in] 양수 및 음수 값을 사용하여 설정할 수 있는 글꼴 크기. 양수 값의 경우 표시되는 텍스트의 크기는 운영 체제의 글꼴 크기 설정에 따라 달라지지 않습니다. 음수 값의 경우 이 값은 점의 10분의 1로 설정되며 텍스트 크기는 운영 체제 설정("표준 척도" 또는 "대 척도")에 따라 달라집니다. 모드 간의 차이에 대한 자세한 내용은 아래 참고 사항을 참조하십시오.

flags

[in] 폰트 스타일을 설명하는 플래그 조합.

orientation

[in] X축에 대한 텍스트의 수평 기울기, 측정 단위는 0.1도입니다. 즉, *orientation=450*은 45도 기울기를 나타냅니다.

반환값

현재 글꼴이 성공적으로 설치된 경우 true를 반환하고, 그렇지 않은 경우 false를 반환합니다. 가능한 코드 에러:

- ERR_INVALID_PARAMETER(4003) - *name* presents NULL or "" (empty string),
- ERR_INTERNAL_ERROR(4001) – 운영 체제 오류(예를 들어, 존재하지 않는 글꼴 만들기 시도).

참고

글꼴 이름에 "::"가 사용되는 경우, 글꼴은 EX5 리소스에서 다운로드됩니다. *이름* 글꼴명이 확장명으로 지정되면, 파일에서 글꼴이 다운로드 되고, 경로가 "₩ 또는 "/"에서 시작하면, 파일은 MQL5 디렉토리를 기준으로 검색됩니다. 그렇지 않으면, TextSetFont() 함수라고 하는 EX5 파일의 경로를 기준으로 검색됩니다.

글꼴 크기는 양수 또는 음수 값을 사용하여 설정합니다. 이 점은 운영 체제 설정(크기 척도)에서 텍스트 크기의 종속성을 정의합니다.

- 크기가 양수를 사용하여 지정된 경우 이 크기는 논리 글꼴을 실제 글꼴로 변경할 때 장치의 물리적 측정 단위(픽셀)로 변환되며, 이 크기는 사용 가능한 글꼴에서 선택한 심볼 글리프의 높이에 해당합니다. 이 경우는 TextOut() 함수로 표시되는 텍스트와 OBJ_LABEL ("레이블") 그래픽 객체에 의해 표시되는 텍스트를 차트에서 함께 사용하는 것이 좋습니다.
- 음수를 사용하여 크기를 지정한 경우 이 숫자는 논리 포인트의 10분의 1로 설정되고 10으로 분할됩니다(예: -350은 논리 점 35과 동일함). 획득한 값은 장치의 물리적 측정 단위(픽셀)로 변환되며 사용

© 2000-2025, MetaQuotes Ltd.

가능한 글꼴에서 선택한 심볼 높이의 절대 값에 해당합니다. 개체 속성에 지정된 글꼴 크기에 -10을 곱하면 화면의 텍스트 크기가 OBJ_LABEL 개체의 텍스트 크기와 비슷해집니다.

플래그를 글꼴 너비를 지정하는 플래그 중 하나와 스타일 플래그의 조합으로 사용할 수 있습니다. 아래에 플래그 이름이 나와 있습니다.

글꼴 스타일 지정 플래그

플래그	설명
FONT_ITALIC	이탤릭
FONT_UNDERLINE	언더라인
FONT_STRIKEOUT	취소선

글꼴 너비를 지정하기 위한 플래그

플래그
FW_DONTCARE
FW_THIN
FW_EXTRALIGHT
FW_ULTRALIGHT
FW_LIGHT
FW_NORMAL
FW_REGULAR
FW_MEDIUM
FW_SEMIBOLD
FW_DEMIBOLD
FW_BOLD
FW_EXTRABOLD
FW_ULTRABOLD
FW_HEAVY
FW_BLACK

참고 항목

Resources, ResourceCreate(), ResourceSave(), TextOut()

© 2000-2025, MetaQuotes Ltd.

TextOut

이 함수는 사용자 지정 배열(버퍼)에 텍스트를 표시하고 해당 작업의 결과를 반환합니다. 이 배열은 그래픽 리소스를 생성하도록 설계되었습니다.

```
bool  TextOut(
   const string       text,          // 표시된 텍스트
   int                x,             // X 좌표
   int                y,             // Y 좌표
   uint               anchor,        // 앵커 유형
   uint               &data[],       // 출력 버퍼
   uint               width,         // 버퍼 너비(픽셀)
   uint               height,        // 버퍼 높이(픽셀)
   uint               color,         // 텍스트 색상
   ENUM_COLOR_FORMAT  color_format   // 출력용 색상 형식
   );
```

Parameter

텍스트

[in] 버퍼에 기록될 텍스트를 표시합니다. 한 줄짜리 텍스트만 표시됩니다.

x

[in] 표시된 텍스트의 앵커 포인트의 X 좌표입니다.

y

[in] 표시된 텍스트의 앵커 포인트의 Y 좌표입니다.

anchor

[in] 표시된 텍스트의 앵커 포인트 위치의 9가지 사전 정의된 메소드 중 값입니다. 이 값은 수평 및 수직 텍스트 정렬 플래그의 두 개의 플래그 조합으로 설정됩니다. 플래그 이름은 아래 참고에 나와 있습니다.

data[]

[in] 텍스트가 표시되는 버퍼. 버퍼는 그래픽 리소스를 만드는 데 사용됩니다.

width

[in] 버퍼 너비(픽셀)

height

[in] 버퍼 높이(픽셀)

color

[in] 텍스트 색상.

color_format

[in] 색상 형식은 ENUM_COLOR_FORMAT 열거값으로 설정됩니다.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다.

참고

© 2000-2025, MetaQuotes Ltd.

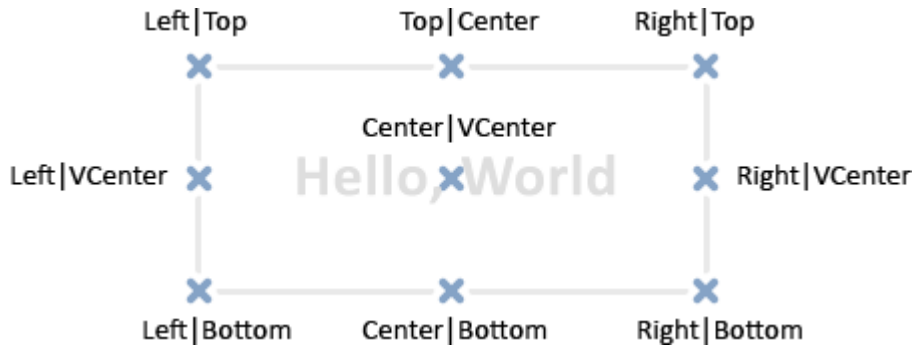
*앵커*에 의해 지정된 앵커 포인트는 수평 및 수직 텍스트 정렬의 두 플래그의 조합입니다. 수평 텍스트 정렬 플래그:

- TA_LEFT – 경계 상자의 왼쪽에 있는 고정점
- TA_CENTER – 수평 앵커 포인트는 경계 상자의 중앙에 위치합니다
- TA_RIGHT – 경계 상자의 오른쪽에 있는 고정점

수직 텍스트 정렬 플래그:

- TA_TOP – 경계 상자의 윗면에 있는 앵커 포인트
- TA_VCENTER – 수직 앵커 포인트는 경계 상자의 중앙에 위치합니다.
- TA_BOTTOM – 경계 상자 아래쪽의 앵커 포인트

플래그와 지정된 앵커 포인트의 가능한 조합이 영상에 표시됩니다.



예:

```
//--- 너비와 캔버스(그리기에 사용되는)의 높이.
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- 스크립트를 시작하기 전에 매개 변수 창을 표시합니다
#property script_show_inputs
//--- 색상 형식을 설정합니다
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- 그리기 배열(버퍼)
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그리기를 위해 OBJ_BITMAP_LABEL 개체 생성
   ObjectCreate(0,"CLOCK",OBJ_BITMAP_LABEL,0,0,0);
//--- CLOCK 객체에 쓸 그래픽 리소스의 이름을 명시합니다
   ObjectSetString(0,"CLOCK",OBJPROP_BMPFILE,"::IMG");

//--- auxiliary variables
   double a;            // 애로우 코너
   uint   nm=2700;      // 분(minute) 코너
   uint   nh=2700*12;   // 시간(hour) 카운터
   uint   w,h;          // 텍스트 문자열 크기를 수신하기 위한 변수
```

© 2000-2025, MetaQuotes Ltd.

```
   int     x,y;           // 텍스트 문자열 앵커 지점의 현재 좌표를 계산하기 위한 변수

//--- 스크립트가 중지될 때까지 시계 바늘을 무한 루프에서 회전합니다
   while(!IsStopped())
     {
      //--- 시계 그리기 버퍼 배열을 지웁니다
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- 시계 페이스 숫자 그리기 글꼴 설정
      TextSetFont("Arial",-200,FW_EXTRABOLD,0);
      //--- 시계 페이스 그리기
      for(int i=1;i<=12;i++)
        {
         //--- 시계 페이스의 현재 시간에 대한 크기를 받음
         TextGetSize(string(i),w,h);
         //--- 시계 페이스의 현재 시간 좌표를 계산
         a=-((i*300)%3600*M_PI)/1800.0;
         x=IMG_WIDTH/2-int(sin(a)*80+0.5+w/2);
         y=IMG_HEIGHT/2-int(cos(a)*80+0.5+h/2);
         //--- 시계 페이스의 시간을 ExtImg[] 버퍼로 출력
         TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,
        }
      //--- 이제, 분침을 그릴 글꼴을 지정합니다
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
      //--- 분침의 크기를 받습니다
      TextGetSize("----->",w,h);
      //--- 시계 페이스의 분침의 좌표를 계산
      a=-(nm%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- ExtImg[] 버퍼의 시계 페이스 분침 출력
      TextOut("----->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_f

      //--- 자, 분침을 그릴 폰트를 설정하세요
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
      TextGetSize("==>",w,h);
      //--- 시계 페이스의 시침의 좌표를 계산
      a=-(nh/12%3600*M_PI)/1800.0;
      x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
      y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
      //--- ExtImg[] 버퍼에 대한 시계 페이스의 시침 출력
      TextOut("==>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_forn

      //--- update the graphical resource
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //--- 포스 차트 업데이트
      ChartRedraw();

      //--- 시간 및 분 카운터 증가
      nm+=60;
      nh+=60;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 프레임 사이에 잠시 멈춤
      Sleep(10);
     }
//--- 스크립트 작업을 완료할 때 CLOCK 개체 삭제
   ObjectDelete(0,"CLOCK");
//---
  }
```

참고 항목

Resources, ResourceCreate(), ResourceSave(), TextGetSize(), TextSetFont()

© 2000-2025, MetaQuotes Ltd.

TextGetSize

이 함수는 현재 폰트 설정에서 선 너비 및 높이를 반환합니다.

```
bool  TextGetSize(
   const string       text,          // 텍스트 문자열
   uint&               width,        // 버퍼 너비(픽셀)
   uint&               height        // 버퍼 높이(픽셀)
   );
```

Parameter

텍스트

[in] 길이와 너비를 구해야 하는 문자열.

width

[out] 폭 수신에 대한 입력 매개 변수.

height

[out] 높이 수신에 대한 입력 매개 변수.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 가능한 코드 에러:

- ERR_INTERNAL_ERROR(4001) - 운영 체제 에러.

참고 항목

Resources, ResourceCreate(), ResourceSave(), TextSetFont(), TextOut()

© 2000-2025, MetaQuotes Ltd.

기술 지표 기능

iMA, iAC, iMACD, iIchimoku 등과 같은 모든 기능은 클라이언트 터미널 글로벌 캐시에 해당하는 기술 지표의 복사본을 생성합니다. 이러한 매개변수를 가진 지표 복사본이 이미 있으면 새 복사본이 생성되지 않고 기존 복사본에 대한 참조 카운터가 증가합니다.

이러한 기능들은 해당 지표 등 사본의 핸들을 반환합니다. 또한, 이 핸들을 사용하여 해당 지표로 계산된 데이터를 수신할 수 있습니다. 해당 버퍼 데이터(기술 지표에는 내부 버퍼에 계산된 데이터가 포함되며, 이 데이터는 지표에 따라 1~5까지 다를 수 있음)는 CopyBuffer() 기능을 사용하여 mql5 프로그램에 복사할 수 있습니다.

지표 값을 계산하려면 시간이 필요하므로, 지표 핸들을 OnInit()에서 생성하는 것이 더 좋기 때문에 생성된 직후에는 지표 데이터를 참조할 수 없습니다. iCustom() 함수는 해당 사용자 지정 지표를 생성하고 성공적으로 생성된 경우 해당 핸들을 반환합니다. 사용자 지정 지표는 최대 512개의 지표 버퍼를 포함할 수 있으며, 이 버퍼의 내용은 CopyBuffer() 기능을 통해서도 얻을 수 있습니다.

IndicatorCreate() 기능을 사용하여 기술 지표를 생성하는 보편적인 방법이 있습니다. 이 기능은 다음의 데이트럴 입력 매개변수로 수신합니다:

- 심볼 이름;
- 시간프레임;
- 생성할 지표 유형;
- 지표의 입력 매개변수의 수;
- 필요한 모든 입력 매개변수를 포함하는 MqlParam 유형의 배열.

지표 핸들이 전달되는 IndicatorRelease() 기능을 사용하여 더 이상 사용되지 않는 지표 등으로부터 컴퓨터 메모리를 분리할 수 있습니다.

노트. mql5 프로그램 내에서 동일한 매개변수를 사용하여 지표 기능을 반복해서 호출해도 기준 카운터가 여러 번 증가하지 않으며 카운터는 1만큼만 증가합니다. 그러나 지표 핸들은 함수 OnInit() 또는 클래스 생성기에서 가져오고 다른 기능에서 추가로 사용하는 것이 좋습니다. 기준 카운터는 mql5 프로그램이 초기화 해제되면 감소합니다.

모든 지표 기능에는 심볼 및 기간의 매개변수가 2개 이상 있습니다. 심볼의 NULL 값은 현재 심볼을 의미하며, 기간의 0 값은 현재 시간 프레임을 의미합니다.

기능	지표의 핸들을 반환:
iAC	Accelerator Oscillator
iAD	누적/분배
iADX	평균 방향 인덱스
iADXWilder	웰즈 와일더의 평균 방향 인덱스
iAlligator	엘리게이터
iAMA	적응형 이동 평균
iAO	어썸 오실레이터
iATR	평균 참값 범위

© 2000-2025, MetaQuotes Ltd.

기능	지표의 핸들을 반환:
iBearsPower	베어즈 파워
iBands	볼링어 밴즈®
iBullsPower	불스 파워
iCCI	상품 채널 인덱스
iChaikin	체이킨 오실레이터
iCustom	고객 지표
iDEMA	이중 지수 이동 평균
iDeMarker	DeMarker
iEnvelopes	봉투
iForce	포스 인덱스
iFractals	프랙탈
iFrAMA	프랙탈 적응형 이동 평균
iGator	게이터 오실레이터
iIchimoku	이치모쿠 킨코 효
iBWMFI	빌 윌리엄즈의 시장 촉진 인덱스
iMomentum	모멘텀
iMFI	자금 흐름 인덱스
iMA	이동 평균
iOsMA	오실레이터의 이동 평균 (MACD 히스토그램)
iMACD	이동 평균 수렴-분산
iOBV	잔고량
iSAR	포물선 정지 및 역 시스템
iRSI	상대 강도 지수
iRVI	상대 활력 지수
iStdDev	표준 편차
iStochastic	스토캐스틱 오실레이터
iTEMA	삼중 지수 이동 평균
iTriX	삼중 지수 이동 평균 오실레이터
iWPR	윌리엄즈의 퍼센트 범위
iVIDyA	변수 인덱스 동적 평균

© 2000-2025, MetaQuotes Ltd.

기능	지표의 핸들을 반환:
iVolumes	볼륨

© 2000-2025, MetaQuotes Ltd.

iAC

이 함수는 클라이언트 터미널의 글로벌 캐시에 가속기 오실레이터를 만들고 해당 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iAC(
   string           symbol,     // 심볼명
   ENUM_TIMEFRAMES  period      // 기간
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 열거 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                    Demo_iAC.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iAC 기술 지표의 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plotting of iAC
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                  |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,               // iAC 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iAC;          // 함수 유형
input string                symbol=" ";             // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 타임프레임
//--- 지표 버퍼
double iACBuffer[];
double iACColors[];
//--- iAC 지표의 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 가속기 오실레이터 지표의 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("심볼 %s/%s에 대한 iAC 지표의 핸들을 생성하지 못했음, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 가속기 오실레이터 지표의 기호/시간 표시
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAC 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표 계산의 첫 번째 시작인지 또는 iAC 지표의 값 수가 변경되었는지
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iACBuffer 배열이 기호/주기에 대한 iAC 지표의 값 수보다 크면
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
```

© 2000-2025, MetaQuotes Ltd.

```
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iACBuffer 및 iACColor 어레이를 Accelerator 오실레이터 지표의 값으로 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 가속기 오실레이터 지표의 값 수 외우기
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAC 지표에서 지표 버퍼 채우기                                    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // 가속기 오실레이터 값의 지표 버퍼
                          double &color_indexes[], // 색상 버퍼(색상 인덱스 저장용)
                          int ind_handle,          // iAC 지표 핸들
                          int amount               // 복제 값 수
                          )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iACBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAC 지표에서 데이터를 복사하지 못했습니다, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 이제 색상의 색인을 복제합니다
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAC 지표에서 색상 값을 복사하지 못했습니다, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAD

함수는 축적/분배 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iAD(
   string               symbol,             // 심볼명
   ENUM_TIMEFRAMES      period,             // 주기
   ENUM_APPLIED_VOLUME  applied_volume      // 계산을 위한 볼륨 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 열거 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

applied_volume

[in] 이용된 볼륨. ENUM_APPLIED_VOLUME 값 중 하나일 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAD.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "i AD 기술 지표의 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAD
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,               // iAD 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iAD;          // 함수 유형
input ENUM_APPLIED_VOLUME   volumes;                // 사용된 볼륨
input string                symbol=" ";             // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 타임프레임
//--- 지표 버퍼
double          iADBuffer[];
//--- iAD 지표의 핸들을 저장하기 위한 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 축적/분배 지표의 값 수를 유지합니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("심볼 %s/%s 에 대한 iAD 지표의 핸들을 생성하지 못했음, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 축적/분배 지표의 기호/시간 표시
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAD 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산이 처음 시작되거나 iAD 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iADBuffer 배열이 심볼/주기에 대한 iAD 지표의 값 수보다 크면 모든 항목을 복사하지
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iADBuffer 배열을 축적/분배 지표 값으로 채웁니다.
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 축적/분배 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAD 지표에서 지표 버퍼 채우기                                    |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // 축적/분배 라인의 지표 버퍼
                         int ind_handle,     // iAD 지표 핸들
                         int amount          // 복제 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iADBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAD 지표에서 데이터를 복사하지 못했음, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iADX

이 함수는 Average Directional Movement Index 지표의 핸들을 반환합니다.

```
int  iADX(
   string           symbol,          // 심볼명
   ENUM_TIMEFRAMES  period,          // 기간
   int              adx_period       // 평균기간
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

adx_period

[in] 인덱스를 계산할 기간.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 번호는 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE 입니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iADX.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "i ADX 기술 지표에 대한 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot ADX
#property indicator_label1  "ADX"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // use iADX
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iADX;         // type of the function
input int                   adx_period=14;          // 계산 주기
input string                symbol=" ";             // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 타임프레임
//--- 지표 버퍼
double          ADXBuffer[];
double          DI_plusBuffer[];
double          DI_minusBuffer[];
//--- iADX 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 평균 방향 이동 지수에서 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("심볼 %s/%s에 대한 iADX 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 평균 방향 이동 지표가 계산된 심볼/타임프레임 표시
   short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_perio
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iADX 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산을 처음 시작하는 경우 또는 iADX 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iADX 버퍼 배열이 심볼/주기에 대한 iADX 지표 값 수보다 크면 모든 항목을 복사하지 않
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열을 평균 방향 이동 색인 값으로 채움
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 평균 방향 이동 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iADX 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &adx_values[],      // ADX 라인의 지표 버퍼
                           double &DIplus_values[],   // DI+용 지표 버퍼
```

© 2000-2025, MetaQuotes Ltd.

```
                           double &DIminus_values[],  // DI-용 지표 버퍼
                           int ind_handle,            // iADX 지표 핸들
                           int amount                 // 복제 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iADX 버퍼 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iADX 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- DI_plusBuffer 배열의 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iADX 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- DI_minusBuffer 배열의 일부를 인덱스 2가 있는 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iADX 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iADXWilder

이 함수는 웰즈 와일더의 평균 방향 이동 지수 핸들을 반환합니다.

```
int  iADXWilder(
   string           symbol,          // 심볼명
   ENUM_TIMEFRAMES  period,          // 기간
   int              adx_period       // 평균기간
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

adx_period

[in] 인덱스를 계산할 기간.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 번호는 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE 입니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                  iADXWilder.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iADXWilder 기술 지표의 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot ADX
#property indicator_label1  "ADX"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // iADXWilder 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iADXWilder;   // 함수 유형
input int                   adx_period=14;          // 계산 주기
input string                symbol=" ";             // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 타임프레임
//--- 지표 버퍼
double          ADXBuffer[];
double          DI_plusBuffer[];
double          DI_minusBuffer[];
//--- iADXWilder 지표의 핸들을 저장하기 위한 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 웰즈 와일더의 평균 방향 이동 지수에 있는 값의 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iADXWilder 지표의 핸들을 만들지 못했습니다, 오류 코드 %
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 웰스 와일더의 평균 방향 이동 지표가 계산된 심볼/시간 프레임 표시
   short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),adx
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iADXWilder 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iADXWilder 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iADXFuffer 배열이 심볼/주기에 대한 iADXWilder 지표의 값 수보다 크면 모든 항목을
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열을 웰스 와일더에 의한 평균 방향 이동 지수 값으로 채움
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 평균 방향 이동 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iADXWilder 지표에서 지표 버퍼 채우기                             |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &adx_values[],      // ADX 라인의 지표 버퍼
                           double &DIplus_values[],   // DI+용 지표 버퍼
```

© 2000-2025, MetaQuotes Ltd.

```
                           double &DIminus_values[],  // DI-용 지표 버퍼
                           int ind_handle,            // iADXWilder 지표 핸들
                           int amount                 // 복제 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iADX 버퍼 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iADXWilder 지표에서 데이터를 복사하지 못했습니다, 오류 코드 %d",GetLastEr
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- DI_plusBuffer 배열의 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iADXWilder 지표에서 데이터를 복사하지 못했습니다, 오류 코드 %d",GetLastEr
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- DI_plusBuffer 배열의 일부를 인덱스 2가 있는 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iADXWilder 지표에서 데이터를 복사하지 못했습니다, 오류 코드 %d",GetLastEr
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAlligator

이 기능은 앨리게이터 지표의 핸들을 반환합니다.

```
int  iAlligator(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  jaw_period,       // 턱 계산 기간
   int                  jaw_shift,        // 턱의 수평 이동
   int                  teeth_period,     // 이빨 계산 기간
   int                  teeth_shift,      // 이빨의 수평 이동
   int                  lips_period,      // 입술 계산 기간
   int                  lips_shift,       // 입술 수평 이동
   ENUM_MA_METHOD       ma_method,        // 평활 유형
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol
[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period
[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

jaw_period
[in] 파란색 선의 평균 주기(악어 턱)

jaw_shift
[in] 가격 관리도에 상대적인 파란색 선의 이동.

teeth_period
[in] 빨간색 선의 평균 주기(악어 톱니).

teeth_shift
[in] 가격 차트를 기준으로 한 빨간색 선의 이동.

lips_period
[in] 녹색 선의 평균 주기(악어 입술).

lips_shift
[in] 가격표에 상대적인 녹색선의 이동.

ma_method
[in] 평균 산출 방법. ENUM_MA_METHOD 값 중 하나일 수 있습니다.

applied_price
[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

© 2000-2025, MetaQuotes Ltd.

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 번호는 다음과 같습니다. 0 - GATORJAW_LINE, 1 - GATORTEETH_LINE, 2 - GATORLIPS_LINE.

예제:

```
//+------------------------------------------------------------------+
//|                                                Demo_iAlligator.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iAlligator 기술 지표에 대한 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 표준 악어와 유사합니다."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- plot Jaws
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot Teeth
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot Lips
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   Call_iAlligator,         // iAlligator 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iAlligator;   // 함수 유형
input string                symbol=" ";             // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 타임프레임
input int                   jaw_period=13;          // 턱선 주기
input int                   jaw_shift=8;            // 턱선 이동
input int                   teeth_period=8;         // 이빨선 주기
input int                   teeth_shift=5;          // 이빨선 이동
input int                   lips_period=5;          // 입술선 주기
input int                   lips_shift=3;           // 입술선 이동
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // 앨리게이터 선 평균 산출 방법
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// 앨리게이터 계산에 사용되는 가격 
//--- 지표 버퍼
double          JawsBuffer[];
double          TeethBuffer[];
double          LipsBuffer[];
//--- iAlligator 지표 핸들을 저장하기 위한 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 앨리게이터 지표의 값 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- 각 선의 이동 설정
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iAlligator)
      handle=iAlligator(name,period,jaw_period,jaw_shift,teeth_period,
                        teeth_shift,lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[8];
     //--- 악어 선의 주기 및 이동
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
//--- 평활 유형
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- 가격 유형
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- 핸들 생성
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("심볼 %s/%s에 대한 iAlligator 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 악어 지표가 계산된 심볼/타임프레임 표시
   short_name=StringFormat("iAlligator(%s/%s, %d,%d,%d,%d,%d,%d)",name,EnumToString(pe
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAlligator 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표 계산이 처음 시작되거나 iAlligator 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- JawsBuffer 배열이 심볼/주기에 대한 iAlligator 지표의 값 수보다 크면 모든 항목을
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열을 악어 지시자의 값으로 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment(comm);
//--- 앨리게이터 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAlligator 지표에서 지표 버퍼 채우기                             |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // 턱선용 지표 버퍼
                           int j_shift,            // 턱선 이동
                           double &teeth_buffer[], // 이빨 선 지표 버퍼
                           int t_shift,            // 이빨 선 이동
                           double &lips_buffer[],  // 입술 선 지표 버퍼
                           int l_shift,            // 입술 선 이동
                           int ind_handle,         // iAlligator 지표 핸들
                           int amount              // 복제 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- JawsBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAlligator 지표에서 데이터를 복사하지 못했습니다, 오류 코드 %d",GetLastEr
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- TeethBuffer 배열의 일부를 인덱스 1이 있는 표시기 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAlligator 지표에서 데이터를 복사하지 못했습니다, 오류 코드 %d",GetLastEr
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- LipsBuffer 배열의 일부를 인덱스 2가 있는 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAlligator 지표에서 데이터를 복사하지 못했습니다, 오류 코드 %d",GetLastEr
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAMA

이 함수는 적응형 이동 평균 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iAMA(
   string               symbol,              // 심볼명
   ENUM_TIMEFRAMES      period,              // 주기
   int                  ama_period,          // AMA용 평균 주기
   int                  fast_ma_period,      // 빠른 이동평균 주기
   int                  slow_ma_period,      // 느린 이동평균 주기
   int                  ama_shift,           // 지표의 수평 이동
   ENUM_APPLIED_PRICE   applied_price        // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ama_period

[in] 효율성 계수가 계산되는 계산 기간.

fast_ma_period

[in] 빠른 시장을 위한 평활화 계수 계산의 빠른 기간.

slow_ma_period

[in] 추세가 없는 상태에서 평활 계수 계산의 느린 기간.

ama_shift

[in] 가격 차트에 상대적인 지표 이동.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iAMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iAMA 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 표준 AMA와 유사합니다."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iAMA
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // iAMA 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iAMA;          // 함수 유형
input string                symbol=" ";              // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // 타임프레임
input int                   ama_period=15;           // 계산 주기
input int                   fast_ma_period=2;        // 빠른 MA 주기
input int                   slow_ma_period=30;       // 느린 MA 주기
input int                   ama_shift=0;             // 수평 이동
input ENUM_APPLIED_PRICE    applied_price;           // 가격 유형
//--- 지표 버퍼
double          iAMABuffer[];
//--- iAMA 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 적응형 이동 평균 지표의 값 수를 유지합니다.
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 매핑
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- 시프트 세팅
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,appli
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[5];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ama_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=fast_ma_period;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slow_ma_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ama_shift;
      //--- 가격 유형
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iAMA 지표의 핸들 생성 실패, , 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 적응형 이동 평균 지표의 심볼/타임프레임 표시
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAMA 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산을 처음 시작하는 경우 또는 iAMA 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iAMA 버퍼 배열이 심볼/주기에 대한 iAMA 지표의 값 수보다 크면 모든 항목을 복사하지
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열을 적응형 이동 평균 지표의 값으로 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment(comm);
//--- 적응형 이동 평균 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAMA 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // AMA 선 지표 버퍼
                         int a_shift,           // AMA 선 이동
                         int ind_handle,        // iAMA 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iAMABuffer 배열의 일부를 인덱스가 0인 표시기 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAMA 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iAO

이 함수는 놀라운 오실레이터 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iAO(
   string           symbol,     // 심볼명
   ENUM_TIMEFRAMES  period      // 기간
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAO.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "i AO 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- the iAO plot
#property indicator_label1  "iAO"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen,clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
enum Creation
  {
   Call_iAO,               // iAO 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iAO;          // 함수 유형
input string                symbol=" ";             // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 타임프레임
//--- 지표 버퍼
double          iAOBuffer[];
double          iAOColors[];
//--- iAO 지표의 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- Awesome 오실레이터 지표 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iAO 지표 핸들 생성 실패, 오류 코드 %d",
```

© 2000-2025, MetaQuotes Ltd.

```
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Awesome 오실레이터 지표가 계산된 심볼/타임프레임 표시
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iAO 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산을 처음 시작하는 경우 또는 iAO 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iAOBuffer 배열이 기호/주기에 대한 iAO 지표 값 수보다 크면 모든 항목을 복사하지 않
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- iAOBuffer 및 iAOColors 배열을 Awesome 오실레이터 지표의 값으로 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Awesome 오실레이터 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAO 지표에서 지표 버퍼 채우기                                    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // Awesome 오실레이터 값의 지표 버퍼
                          double &color_indexes[], // 색상 버퍼(색상 인덱스 저장용)
                          int ind_handle,          // iAO 지표 핸들
                          int amount               // 복제 값 수
                          )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iAOBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채움
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAO 지표에서 데이터 복사 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 이제 색상의 색인을 복제합니다
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iAO 지표에서 색상 값 복사 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iATR

이 함수는 평균 실질 범위(Average True Range) 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iATR(
   string           symbol,        // 심볼명
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균 주기
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 지표 계산을 위한 평균 기간의 값.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iATR.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iATR 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// iATR 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input int                  atr_period=14;          // 계산 주기
input Creation             type=Call_iATR;         // 함수 유형
input string               symbol=" ";             // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // 타임프레임
//--- 지표 버퍼
double          iATRBuffer[];
//--- iAC 지표의 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- Average True Range(평균 실질 범위) 지표의 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iATR 지표의 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 평균 실질 범위 지표가 계산된 심볼/타임프레임 표시
   short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iATR 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산이 처음 시작되거나 iATR 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iATRBuffer 배열이 심볼/주기 동안 iATR 지표의 값 수보다 크면 모든 항목을 복사하지
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iATRBuffer 배열을 Average True Range 지표의 값으로 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 평균 실질 범위 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iATR 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // ATR 값에 대한 지표 버퍼
                         int ind_handle,    // iATR 지표 핸들
                         int amount         // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iATRBuffer 배열의 일부를 인덱스가 0인 표시기 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iATR 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBearsPower

이 기능은 베어즈 파워(Bears Power) 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iBearsPower(
   string               symbol,            // 심볼명
   ENUM_TIMEFRAMES      period,            // 기간
   int                  ma_period,         // 평균 주기
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 지표 계산을 위한 평균 기간의 값.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBearsPower.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iBearsPower 기술 지표에 대한 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iBearsPower plot
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,       // iBearsPower 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iBearsPower;  // 함수 유형
input int                   ma_period=13;           // 이동 평균 주기
input string                symbol=" ";             // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 타임프레임
//--- 지표 버퍼
double          iBearsPowerBuffer[];
//--- iBearsPower 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 우리는 Bears Power 지표의 값 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[1];
      //--- period of ma
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iBearsPower 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Bears Power 지표가 계산된 심볼/타임프레임 표시
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),m
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iBearsPower 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iBearsPower 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iBearsPowerBuffer 배열이 심볼/주기 동안 iBearsPower 지표 값 수보다 크면 모든 항
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iBearsPowerBuffer 배열을 Bears Power 지표 값으로 채움
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 베어스 파워 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBearsPower 지표에서 지표 버퍼 채우기                            |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // 베어즈 파워 값에 대한 지표 버퍼
                         int ind_handle,    // iBearsPower 지표 핸들
                         int amount         // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iBearsPowerBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iBearsPower 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBands

이 함수는 볼린저 밴드® 지표의 핸들을 반환합니다.

```
int  iBands(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  bands_period,     // 평균 선 계산 주기
   int                  bands_shift,      // 지표의 수평 이동
   double               deviation,        // 표준 편차 수
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

bands_period

[in] 지표선 주기의 평균 주기.

bands_shift

[in] 가격 차트를 기준으로 지표를 이동.

deviation

[in] 주요 선으로부터의 편차.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 번호는 다음과 같습니다: 0 - BASE_LINE, 1 - UPPER_BAND, 2 - LOWER_BAND

예제:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iBands.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iBands 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- 상단 플롯
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 하단 플롯
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- 중단 플롯
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,             // iBands 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iBands;          // 함수의 종류
input int                   bands_period=20;           // 이동 평균 주기
input int                   bands_shift=0;             // 이동
input double                deviation=2.0;             // 표준편차 수
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 가격 유형
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          UpperBuffer[];
double          LowerBuffer[];
double          MiddleBuffer[];
//--- iBands 지표의 핸들을 저장하기 위한 변수
int     handle;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 볼린저 밴드 지표의 값 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- 각 선의 이동 설정
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[4];
      //--- ma 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- 이동
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- 표준편차의 수
      pars[2].type=TYPE_DOUBLE;
      pars[2].double_value=deviation;
      //--- 가격 유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
```

© 2000-2025, MetaQuotes Ltd.

```
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iBands 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 볼린저 밴드 지표의 심볼/타임프레임 표시
   short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
                           bands_period,bands_shift,deviation,EnumToString(applied_pri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iBands 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iBands 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- 지표 버퍼의 크기가 심볼/주기 iBands 지표 값 수보다 크면 모든 항목을 복사하지 않습니
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
```

© 2000-2025, MetaQuotes Ltd.

```
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열을 볼린저 밴드 지표 값으로 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,v
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 볼린저 밴드 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBands 지표 지표 버퍼 채우기                                     |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // 볼린저 밴드의 중간선 표시 버퍼
                           double &upper_values[],    // 위쪽 테두리의 지표 버퍼
                           double &lower_values[],    // 아래쪽 테두리의 지표 버퍼
                           int shift,                 // 이동
                           int ind_handle,            // iBands 지표 핸들
                           int amount                 // 복제 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- 인덱스가 0인 지표 버퍼의 값으로 MiddleBuffer 배열의 일부를 채웁니다
   if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iBands 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- UpperBuffer 배열의 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iBands 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- LowerBuffer 배열의 일부를 인덱스 2가 있는 지표시기 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iBands 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBullsPower

이 기능은 불스 파워(Bulls Power) 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iBullsPower(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 주기
   int                  ma_period,        // 평균 주기
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 지표 계산에 대한 평균 주기.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBullsPower.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iBullsPower 기술 지표에 대한 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iBullsPower plot
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,       // iBullsPower 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iBullsPower;  // 함수 유형
input int                   ma_period=13;           // 이동 평균 주기
input string                symbol=" ";             // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // 타임프레임
//--- 지표 버퍼
double          iBullsPowerBuffer[];
//--- iBullsPower 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- Bulls Power 지표에 값 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[1];
      //--- ma 주기
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iBullsPower 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 불스 파워(Bulls Power) 지표가 계산된 기호/타임프레임을 표시합니다
   short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iBullsPower 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iBullsPower 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iBullsPowerBuffer 배열이 기호/주기 동안 iBullsPower 지표 값 수보다 크면 모든 항
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iBullsPowerBuffer 배열을 BullsPower 지표 값으로 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Bulls Power 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBullsPower 지표에서 지표 버퍼 채우기                            |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Bulls Power 값의 지표 버퍼
                         int ind_handle,    // iBullsPower 지표 핸들
                         int amount         // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iBullsPowerBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iBullsPower 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

iCCI

이 함수는 상품 채널 색인 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iCCI(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  ma_period,        // 평균 주기
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 지표 계산의 평균 주기.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iCCI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iCCI 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iCCI plot
#property indicator_label1  "iCCI"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- horizontal levels in the indicator window
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // iCCI 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iCCI;              // 함수 유형
input int                   ma_period=14;                // 이동 평균 주기
input ENUM_APPLIED_PRICE    applied_price=PRICE_TYPICAL; // 가격 유형
input string                symbol=" ";                  // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // 타임프레임
//--- 지표 버퍼
double          iCCIBuffer[];
//--- iCCI 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 상품 채널 색인 지표 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 지표 핸들 생성
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[2];
      //--- 이동 평균 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 가격 유형
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iCCI 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- CCI 지표가 계산된 심볼/타임프레임을 표시
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iCCI 지표에서 복사된 값 수
   int values_to_copy;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산이 처음 시작되거나 iCCI 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iCCI 버퍼 배열이 기호/주기에 대한 iCCI 지표 값 수보다 크면 모든 항목을 복사하지 않
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iCCIBuffer 배열을 물품 채널 색인 표시기 값으로 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 상품 채널 색인 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iCCI 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // 상품 채널 색인 값의 지표 버퍼
                         int ind_handle,    // iCCI 지표 핸들
                         int amount         // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iCCBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("iCCI 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
       return(false);
      }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iChaikin

이 함수는 Chaikin 오실레이터 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iChaikin(
   string               symbol,              // 심볼명
   ENUM_TIMEFRAMES      period,              // 주기
   int                  fast_ma_period,      // 빠른 주기
   int                  slow_ma_period,      // 느린 주기
   ENUM_MA_METHOD       ma_method,           // 평활 유형
   ENUM_APPLIED_VOLUME  applied_volume       // 볼륨 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

fast_ma_period

[in] 계산을 위한 빠른 평균화 주기.

slow_ma_period

[in] 계산을 위한 느린 평균화 주기.

ma_method

[in] 평활 유형. ENUM_MA_METHOD의 평균 상수 중 하나일 수 있습니다.

applied_volume

[in] 사용한 볼륨. ENUM_APPLIED_VOLUME 상수 중 하나일 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                Demo_iChaikin.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iChaikin 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iChaikin plot
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,           // iChaikin 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iChaikin;           // 함수 유형
input int                   fast_ma_period=3;             // 빠른 ma 주기
input int                   slow_ma_period=10;            // 느린 ma 주기
input ENUM_MA_METHOD        ma_method=MODE_EMA;           // 평활 유형
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // 볼륨 유형
input string                symbol=" ";                   // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 타임프레임
//--- 지표 버퍼
double          iChaikinBuffer[];
//--- iChaikin 지표 핸들을 저장하기 위한 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- Chaikin 오실레이터 지표 값 수를 유지합니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
```

© 2000-2025, MetaQuotes Ltd.

```
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volu
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[4];
      //--- 빠른 ma 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- 느린 ma 주기
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- 평활유형
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- 볼륨 유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iChaikin 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Chaikin 오실레이터 지표의 심볼/타임프레임 표시
   short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %s)",name,EnumToString(period)
                           fast_ma_period,slow_ma_period,
                           EnumToString(ma_method),EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iChaikin 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산이 처음 시작되거나 iChaikin 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iChaikinBuffer 배열이 기호/주기 동안 iChaikin 지표 값 수보다 크면 모든 항목을 복
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iChaikinBuffer 배열을 Chaikin 오실레이터 지표 값으로 채웁니다.
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Chaikin 오실레이터 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
```

© 2000-2025, MetaQuotes Ltd.

```
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iChaikin 지표에서 지표 버퍼 채우기                               |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Chaikin 오실레이터 값의 지표 버퍼
                         int ind_handle,    // iChaikin 지표 핸들
                         int amount         // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iChaikinBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iChaikin 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iCustom

이 함수는 지정된 커스텀 지표의 핸들을 반환합니다.

```
int  iCustom(
   string           symbol,     // 심볼명
   ENUM_TIMEFRAMES  period,     // 주기
   string           name        // folder/custom_indicator_name
   ...                          // 지표 입력 파라미터 목록
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

name

[in] 사용자 정의 표시기 이름 앞에 백슬래시 '\'가 표시되면 EX5 표시기 파일은 MQL5 루트 폴더를 기준으로 검색됩니다. 따라서iCustom(Symbol(), Period(), "\FirstIndicator"...) 호출의 경우 표시기는 MQL5\Indicators\FirstIndicator.ex5로 로드됩니다. 이 경로에서 파일을 찾을 수 없으면 오류 4802(ERR_INDICATOR_CANNOT_CREATE)가 반환됩니다.

경로가 '\'로 시작하지 않는다면, 지표는 다음에서 검색되고 다운로드 됩니다:

- 먼저 지표 EX5 파일이 호출 프로그램의 EX5 파일이 있는 폴더에서 검색됩니다. 예를 들면, CrossMA.EX5 EA 는 MQL5\Experts\MyExperts 에 위치하고 iCustom 호출 (Symbol(), Period(), "SecondIndicator"...)을 포함합니다. 이러한 경우, 지표는 MQL5\Experts\MyExperts\SecondIndicator.ex5 에서 검색됩니다.
- 지표가 동일한 디렉토리에서 발견되지 않는다면, MQL5\Indicators 지표 루트 디렉토리를 기준으로 검색이 수행됩니다. 다른 말로 하면, MQL5\Indicators\SecondIndicator.ex5 파일에 대한 검색이 수행됩니다. 지표가 여전히 검색되지 않는다면, 함수는 INVALID_HANDLE을 반환하며 4802 오류 (ERR_INDICATOR_CANNOT_CREATE)가 촉발됩니다.

지표의 경로가 서브디렉토리 안에 세팅되어 있다면 (예를 들면, MyIndicators\ThirdIndicator), 검색은 MQL5\Experts\MyExperts\MyIndicators\ThirdIndicator.ex5 안에 처음에 호출된 프로그램 폴더(the EA is located in MQL5\Experts\MyExperts)에서 수행됩니다. 성공적이지 않다면, MQL5\Indicators\MyIndicators\ThirdIndicator.ex5 파일에 대해 검색을 수행합니다. 이중 역슬래시 '\'를 경로에 대한 구분자로 사용하는 것을 잊지마십시오, 예를 들면 다음과 같은 : iCustom(Symbol(), Period(), "MyIndicators\\ThirdIndicator"...)

...

[in] 콤마로 구분된 커스텀 지표의 input-parameters. 매개 변수의 유형과 순서가 일치해야 합니다. 지정된 매개변수가 없는 경우 기본값이 사용됩니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

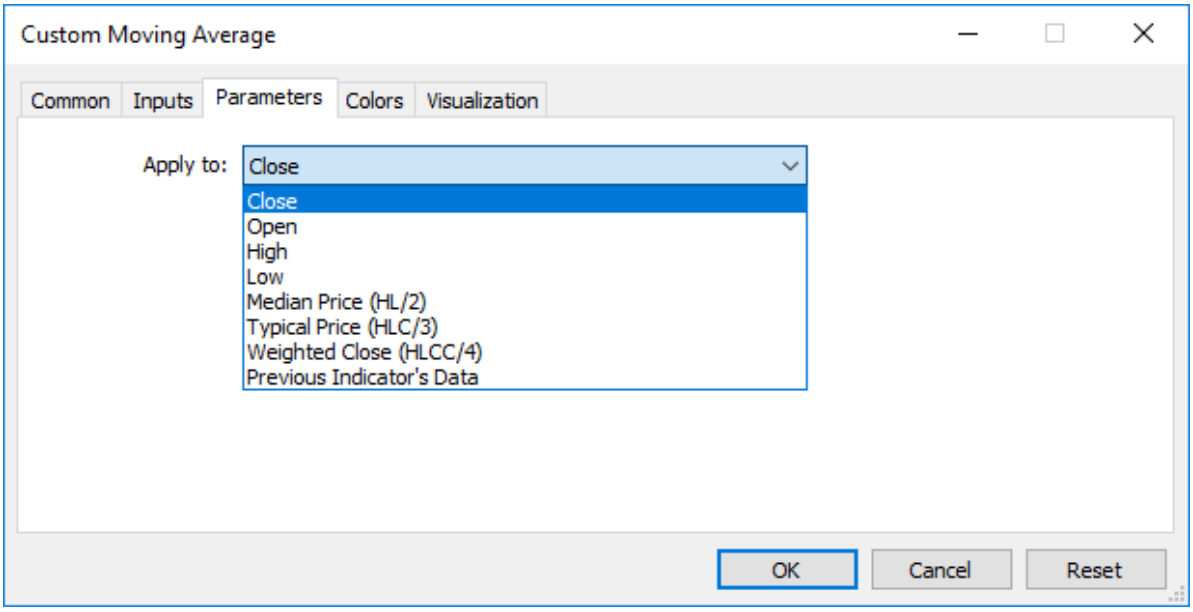
참고

커스텀 지표는 (EX5 확장자와 함께) 컴파일되어야 하며 클라이언트 터미널의 MQL5/Indicators 또는 이 하위 디렉토리에 있어야 합니다.

테스트가 필요한 지표는 iCustom() 함수의 호출로부터 자동으로 정의됩니다, 만약 해당하는 매개변수가 상수 문자열을 통해 세팅된다면. 다른 모든 경우에 대해서는 (지표명을 설정하는 매개변수에서의 IndicatorCreate() 함수의 사용 또는 비-상수 문자열의 사용) #property tester_indicator 속성이 요구됩니다:

```
#property tester_indicator "indicator_name.ex5"
```

지표에서 첫 호출 형태가 쓰였다면, 커스텀 지표 시작시 "매개 변수" 탭에서 연산할 데이터를 추가로 표시할 수 있습니다. "적용 대상" 매개변수를 명시적으로 선택하지 않은 경우 기본 연산은 "종가" 값을 기반으로 합니다.



mql5-프로그램에서 커스텀 지표를 호출할 때 커스텀 지표의 모든 입력 변수 뒤에 Applied_Price 매개변수 또는 다른 지표의 핸들을 마지막으로 전달해야 합니다.

더보기

프로그램 속성, 시계열 및 지표 액세스,IndicatorCreate(), IndicatorRelease()

예제:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Label1
#property indicator_label1  "Label1"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 입력 매개변수
input int MA_Period=21;
```

© 2000-2025, MetaQuotes Ltd.

```
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- 지표 버퍼
double         Label1Buffer[];
//--- 커스텀 Moving Average.mq5 커스텀 지표 핸들
int MA_handle;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼 매핑
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // 종가 사용
                     );
   Print("MA_handle = ",MA_handle,"  error = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- 우리 지표 버퍼에 커스텀 이동 평균 지표의 값을 복사
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy = ",copy,"    rates_total = ",rates_total);
//--- 시도가 실패했다면 - 이를 알려주세요
   if(copy<=0)
      Print("커스텀 이동 평균에 실패한 경우 값을 가져오려는 시도");
//--- 다음 호출에 대한 prev_calculated 값 반환
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

iDEMA

함수는 이중 지수 이동 평균 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iDEMA(
   string               symbol,            // 심볼명
   ENUM_TIMEFRAMES      period,            // 기간
   int                  ma_period,         // 평균 주기
   int                  ma_shift,          // 수평 이동
   ENUM_APPLIED_PRICE   applied_price      // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 계산을 위한 평균 주기(막대 수).

ma_shift

[in] 가격 차트에 상대적인 지표 이동.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iDEMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iDEMA 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_chart_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- the iDEMA plot
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,             // iDEMA 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iDEMA;           // 함수 유형
input int                   ma_period=14;              // 이동평균 주기
input int                   ma_shift=0;                // 이동
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 가격 유형
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          iDEMABuffer[];
//--- iDEMA 지표의 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 이중 지수 이동 평균 지표의 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //--- 시프트 설정
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 지표가 부착된 차트의 심볼을 취합니다
       name=_Symbol;
      }
//--- 지표 핸들 생성
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
      {
       //--- 구조물을 지표의 매개변수로 채움
       MqlParam pars[3];
       //--- 이동 평균 주기
       pars[0].type=TYPE_INT;
       pars[0].integer_value=ma_period;
       //--- 이동
       pars[1].type=TYPE_INT;
       pars[1].integer_value=ma_shift;
       //--- 가격 유형
       pars[2].type=TYPE_INT;
       pars[2].integer_value=applied_price;
       handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
      }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
      {
       //--- 실패에 대해 구분하고 오류 코드를 출력
       PrintFormat("%s/%s 심볼에 대한 iDEMA 지표 핸들 생성 실패, 오류 코드 %d",
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- 지표가 일찍 중단됨
       return(INIT_FAILED);
      }
//--- 이중 지수 이동 평균 지표가 계산된 심볼/타임프레임 표시
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iDEMA 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산을 처음 시작하는 경우 또는 iDEMA 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iDEMA 버퍼 배열이 심볼/주기에 대한 iDEMA 지표의 값 수보다 크면 모든 항목을 복사하지
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iDEMABuffer 배열을 이중 지수 이동 평균 지표의 값으로 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 이중 지수 이동 평균 지표의 값 수 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iDEMA 지표에서 지표 버퍼 채우기                                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // 이중 지수 이동 평균 값의 지표 버퍼
                         int shift,         // 시프트
```

© 2000-2025, MetaQuotes Ltd.

```
                           int ind_handle,     // iDEMA 지표 핸들
                           int amount          // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iDEMABuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iDEMA 지표에서 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iDeMarker

이 함수는 DeMarker 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iDeMarker(
   string           symbol,        // 심볼명
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균 주기
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 계산을 위한 평균 주기(막대 수).

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                Demo_iDeMarker.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iDeMarker 기술 지표에 대한 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iDeMarker plot
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//--- horizontal levels in the indicator window
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,          // iDeMarker 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iDeMarker;       // 함수 유형
input int                   ma_period=14;              // 이동평균 주기
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          iDeMarkerBuffer[];
//--- iDeMarker 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- DeMarker 지표의 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iDeMarker)
      handle=iDeMarker(name,period,ma_period);
   else
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[1];
      //--- 이동 평균 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iDeMarker 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- DeMarker 지표가 계산된 심볼/타임프레임을 표시
   short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iDeMarker 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iDeMarker 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iDeMarkerBuffer 배열이 심볼/주기 동안 iDeMarker 지표의 값 수보다 크면 모든 항목을
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iDeMarkerBuffer 배열을 DeMarker 지표 값으로 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- DeMarker 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iDeMarker 지표에서 지표 버퍼 채우기                              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // DeMarker 값에 대한 지표 버퍼
                         int ind_handle,    // iDeMarker 지표 핸들
                         int amount         // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iDeMarkerBuffer 배열 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iDeMarker 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iEnvelopes

이 함수는 Envelopes 지표의 핸들을 반환합니다.

```
int  iEnvelopes(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  ma_period,        // 평균 선 계산 기간
   int                  ma_shift,         // 지표의 수평 이동
   ENUM_MA_METHOD       ma_method,        // 평활 유형
   ENUM_APPLIED_PRICE   applied_price,    // 핸들 가격 유형
   double               deviation         // 중앙선으로부터의 경계 편차(백분율)
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 메인 선의 평균 주기 평균.

ma_shift

[in] 가격 차트를 기준으로 한 지표의 이동.

ma_method

[in] 평활 유형. ENUM_MA_METHOD 값 중 하나일 수 있습니다.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

deviation

[in] 메인 선으로부터의 편차(백분율).

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 수: 0 -UPPER_LINE, 1 -LOWER_LINE.

예제:

```
//+------------------------------------------------------------------+
//|                                                Demo_iEnvelopes.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iEnvelopes 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- 상단 플롯
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 하단 플롯
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,        // iEnvelopes 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation             type=Call_iEnvelopes;      // 함수 유형
input int                  ma_period=14;              // 이동평균 주기
input int                  ma_shift=0;                // 시프트
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // 평활 유형
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // 가격 유형
input double               deviation=0.1;             // 이동 평균으로부터의 경계 편차
input string               symbol=" ";                // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double         UpperBuffer[];
double         LowerBuffer[];
//--- iEnvelopes 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 차트의 지표명
string short_name;
//--- Envelopes 지표 값 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- 각 선의 이동 설정
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviati
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[5];
      //--- 이동 평균 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 이동
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 평활유형
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- 가격 유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- 가격 유형
      pars[4].type=TYPE_DOUBLE;
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iEnvelopes 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Envelopes 지표가 계산된 심볼/타임프레임을 표시합니다.
   short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %G)",name,EnumToString(pe
   ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iEnvelopes 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 계산을 처음 시작하는 경우 또는 iEnvelopes 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- Upper Buffer 배열이 심볼/주기에 대한 iEnvelopes 지표 값 수보다 크면 모든 항목을 복
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
```

© 2000-2025, MetaQuotes Ltd.

```
      else                          values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- UpperBuffer 및 LowerBuffer 배열을 Envelopes 지표 값으로 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Envelopes 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iEnvelopes 지표에서 지표 버퍼 채우기                             |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],    // 위쪽 테두리의 지표 버퍼
                           double &lower_values[],    // 아래쪽 테두리의 지표 버퍼
                           int shift,                 // 이동
                           int ind_handle,            // iEnvelopes 지표 핸들
                           int amount                 // 복제 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- UpperBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iEnvelopes 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- LowerBuffer 배열의 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iEnvelopes 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                  |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iForce

이 함수는 Force 지수 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iForce(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  ma_period,        // 평균 주기
   ENUM_MA_METHOD       ma_method,        // 평활 유형
   ENUM_APPLIED_VOLUME  applied_volume    // 연산을 위한 볼륨 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 지표 계산의 평균 기간.

ma_method

[in] 평활 유형. ENUM_MA_METHOD 값 중 하나일 수 있습니다.

applied_volume

[in] 사용한 볼륨. ENUM_APPLIED_VOLUME 값 중 하나일 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iForce.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iForce 기술 지표 버퍼의 크기."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- drawing iForce
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,             // iForce 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iForce;             // 함수 유형
input int                   ma_period=13;                 // 평균 주기
input ENUM_MA_METHOD        ma_method=MODE_SMA;           // 평활 유형
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // 볼륨 유형
input string                symbol=" ";                   // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 타임프레임
//--- 지표 버퍼
double          iForceBuffer[];
//--- iForce 지표의 핸들을 저장하는 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- Force 지표 값 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
```

© 2000-2025, MetaQuotes Ltd.

```
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[3];
      //--- 이동 평균 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 평활유형
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_method;
      //--- 볼륨 유형
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_volume;
      //--- 가격 유형
      handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iForce 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Force 지표가 계산되는 심볼/타임프레임을 표시
   short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                           ma_period,EnumToString(ma_method),EnumToString(applied_volu
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iForce 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iForce 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iForceBuffer 배열이 iForce 지표의 심볼/주기 값 수보다 크면 모든 것을 복사하지 않
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iForceBuffer 배열에 Force 지표 값을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Force 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iForce 지표에서 지표 버퍼 채우기                                 |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Force 인덱스 값의 지표 버퍼
                         int ind_handle,    // iForce 지표 핸들
                         int amount         // 복제된 값 수
```

© 2000-2025, MetaQuotes Ltd.

```
                              )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iForceBuffer 배열 일부를 0 인덱스를 가진 지표 버퍼 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iForce 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iFractals

함수는 프랙탈 지표의 핸들을 반환합니다.

```
int  iFractals(
   string           symbol,     // 심볼명
   ENUM_TIMEFRAMES  period      // 기간
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 넘버는 다음과 같습니다: 0 -UPPER_LINE, 1 -LOWER_LINE.

예제:

```
//+------------------------------------------------------------------+
//|                                                Demo_iFractals.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iFractals 기술 지표용 지표 버퍼의 크기."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- the FractalUp plot
#property indicator_label1  "FractalUp"
#property indicator_type1   DRAW_ARROW
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_color1  clrBlue
//--- the FractalDown plot
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,         // iFractals 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iFractals;          // 함수 유형
input string                symbol=" ";                   // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 타임프레임
//--- 지표 버퍼
double          FractalUpBuffer[];
double          FractalDownBuffer[];
//--- iFractals 지표핸들을 저장하는 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 프랙탈 지표 값의 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- set codes using a symbol from the Wingdings charset for the PLOT_ARROW property
   PlotIndexSetInteger(0,PLOT_ARROW,217); // 위 화살표
   PlotIndexSetInteger(1,PLOT_ARROW,218); // 아래 화살표
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 지표 핸들 생성
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iFractals 지표 핸들 생성 실패 , 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 프랙탈 지표가 계산되는 심볼/타임프레임을 표시
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iFractals 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 연산 시작이거나 iFractals 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- FractalUpBuffer 배열이 심볼/주기에 대한 iFractals 지표 값 수보다 크면, 모든 것을
       //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
       //--- 두 개 이하의 막대가 추가되었음을 의미합니다
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- FractalUpBuffer 및 FractalDownBuffer 배열에 Fractals 지표의 값을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
    if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- 메시지 형성
    string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
//--- 차트에 서비스 메시지 표시
    Comment(comm);
//--- 프랙탈 지표의 값 수를 기억
    bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
    return(rates_total);
  }
//+------------------------------------------------------------------+
//| iFractals 지표에서 지표 버퍼 채우기                              |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // 위쪽 화살표 지표 버퍼
                           double &down_arrows[],      // 아래쪽 화살표 지표 버퍼
                           int ind_handle,             // iFractals 지표 핸들
                           int amount                  // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
    ResetLastError();
//--- FractalUpBuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
    if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("FractalUpBuffer 배열에 대한 iFractals 지표 데이터 복제 실패, 오류 코드 %
                   GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
       return(false);
      }
//--- FractalDownBuffer 배열의 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채웁니다
    if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("Fractal DownBuffer 배열에 대한 iFractals 로부터 데이터 복제 실패, 오류 코
                   GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
       return(false);
      }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iFrAMA

이 함수는 프랙탈 적응 이동 평균 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iFrAMA(
   string               symbol,            // 심볼명
   ENUM_TIMEFRAMES      period,            // 기간
   int                  ma_period,         // 평균 주기
   int                  ma_shift,          // 차트에서의 수평 이동
   ENUM_APPLIED_PRICE   applied_price      // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 지표 계산을 위한 주기(바 카운트).

ma_shift

[in] 가격 차트에서 지표 이동.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iFrAMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iFrAMA에 대한 지표 버퍼의 기술 지표."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_chart_window
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_buffers 1
#property indicator_plots   1
//--- drawing iFrAMA
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,            // iFrAMA 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iFrAMA;             // 함수 유형
input int                   ma_period=14;                 // 평균 주기
input int                   ma_shift=0;                   // 시프트
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // 가격 유형
input string                symbol=" ";                   // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 타임프레임
//--- 지표 버퍼
double          iFrAMABuffer[];
//--- iFrAMA 지표 핸들을 저장하는 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 프랙탈 적응 이동 평균 지표 값의 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- 시프트 세팅
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 지표가 부착된 차트의 심볼을 취합니다
       name=_Symbol;
      }
//--- 지표 핸들 생성
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
      {
       //--- 구조물을 지표의 매개변수로 채움
       MqlParam pars[3];
       //--- 이동 평균 주기
       pars[0].type=TYPE_INT;
       pars[0].integer_value=ma_period;
       //--- 이동
       pars[1].type=TYPE_INT;
       pars[1].integer_value=ma_shift;
       //--- 가격 유형
       pars[2].type=TYPE_INT;
       pars[2].integer_value=applied_price;
       //--- 가격 유형
       handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
      }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
      {
       //--- 실패에 대해 구분하고 오류 코드를 출력
       PrintFormat("%s/%s 심볼에 대한 iFrAMA 지표 핸들 생성 실패, 오류 코드 %d",
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- 지표가 일찍 중단됨
       return(INIT_FAILED);
      }
//--- iFrAMA 지표가 계산되는 심볼/타임프레임 표기
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iFrAMA 지표로부터 복제된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 계산 시작이거나 iFrAMA 지표의 값의 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iFrAMABuffer 배열이 심볼/주기에 대한 iFrAMA 지표 안의 값 수보다 클 경우, 모든 것
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- Fractal 적응 이동 평균 지표의 값과 함께 iFrAMABuffer 배열을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 프랙탈 적응 이동 평균 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iFrAMA 지표로부터 지표 버퍼를 채웁니다                           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Fractal 이동 평균 값의 지표 버퍼
```

© 2000-2025, MetaQuotes Ltd.

```
                           int shift,          // 시프트
                           int ind_handle,     // iFrAMA 지표 핸들
                           int amount          // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- 인덱스 0인 지표 버퍼로부터의 값과 iFrAMABuffer 배열 일부를 일부를 채웁니다
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iFrAMA 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iGator

이 함수는 Gator 지표 핸들을 반환합니다. 오실레이터는 앨리게이터의 파란색 선과 빨간색 선(위 히스토그램) 사이의 차이와 빨간색 선과 녹색 선(아래 히스토그램) 간의 차이를 표시합니다.

```
int  iGator(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  jaw_period,       // 턱 계산을 위한 기간
   int                  jaw_shift,        // 턱 수평 이동
   int                  teeth_period,     // 이빨 계산을 위한 기간
   int                  teeth_shift,      // 이빨 수평 이동
   int                  lips_period,      // 입술을 계산하는 기간
   int                  lips_shift,       // 입술 수평 이동
   ENUM_MA_METHOD       ma_method,        // 평활 유형
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

jaw_period

[in] 파란색 선의 평균 주기(악어 턱).

jaw_shift

[in] 가격 관리도에 상대적인 파란색 선의 이동. 지표 히스토그램의 시각적 이동과는 직접 연결되지 않습니다.

teeth_period

[in] 빨간색 선의 평균 주기(악어 톱니).

teeth_shift

[in] 가격 차트를 기준으로 한 빨간색 선의 이동. 지표 히스토그램의 시각적 이동과는 직접 연결되지 않습니다.

lips_period

[in] 녹색 선의 평균 주기(악어 입술).

lips_shift

[in] 가격 차트와 관련된 녹색 선의 이동. 지표 히스토그램의 시각적 이동과는 직접 연결되지 않습니다.

ma_method

[in] 평활 유형. ENUM_MA_METHOD 값 중 하나일 수 있습니다.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 숫자: 0 - UPPER_HISTOGRAM, 1 - 상단 히스토그램의 색상 버퍼, 2 - LOWER_HISTOGRAM, 3 - 하단 히스토그램의 색상 버퍼.

예제:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iGator.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iGator 기술 지표용 지표 버퍼 크기."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 표준 Gator 오실레이터와 같습니다."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- 드로잉 Gator Up
#property indicator_label1  "GatorUp"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- 드로잉 Gator Down
#property indicator_label2  "GatorDown"
#property indicator_type2   DRAW_COLOR_HISTOGRAM
#property indicator_color2  clrGreen, clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iGator,            // iGator 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
```

```
  };
//--- 입력 매개변수
input Creation             type=Call_iGator;       // 함수 유형
input string               symbol=" ";             // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // 타임프레임
input int                  jaw_period=13;          // 턱선 주기
input int                  jaw_shift=8;            // 턱선 이동
input int                  teeth_period=8;         // 이빨선 주기
input int                  teeth_shift=5;          // 이빨선 이동
input int                  lips_period=5;          // 입술선 주기
input int                  lips_shift=3;           // 입술선 이동
input ENUM_MA_METHOD       MA_method=MODE_SMMA;    // 앨리게이터 선 평균 산출 방법
input ENUM_APPLIED_PRICE   applied_price=PRICE_MEDIAN;// 앨리게이터 계산에 사용되는 가격
//--- 지표 버퍼
double         GatorUpBuffer[];
double         GatorUpColors[];
double         GatorDownBuffer[];
double         GatorDownColors[];
//--- iGator 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 위쪽 및 아래쪽 히스토그램에 대한 이동 값
int shift;
//--- Gator 오실레이터 지표 값 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
  파라미터에 지정된 모든 시프트는 게이터 오실레이터가 그려진 기준으로 악어 지표를 참조합니다!
  그래서 게이터 표시기 자체를 움직이지 않고 엘리게이터 선을 움직입니다,
  어떤 값이 Gator 오실레이터 계산에 사용되는지에 대해!
*/
//--- 위 히스토그램과 아래 히스토그램의 이동을 계산해 봅시다. 이는 턱 선과 이빨 선 사이의 차이와
   shift=MathMin(jaw_shift,teeth_shift);
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//--- 위 히스토그램과 아래 히스토그램의 이동을 계산해 봅시다. 이는 턱 선과 이빨 선 사이의 차이와
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iGator)
      handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
                    lips_period,lips_shift,MA_method,applied_price);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[8];
      //--- 엘리게이터 선의 주기 및 이동
      pars[0].type=TYPE_INT;
      pars[0].integer_value=jaw_period;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=jaw_shift;
      pars[2].type=TYPE_INT;
      pars[2].integer_value=teeth_period;
      pars[3].type=TYPE_INT;
      pars[3].integer_value=teeth_shift;
      pars[4].type=TYPE_INT;
      pars[4].integer_value=lips_period;
      pars[5].type=TYPE_INT;
      pars[5].integer_value=lips_shift;
      //--- 평활유형
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- 가격 유형
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- create handle
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iGator 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 게이터 오실레이터 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iGator(%s/%s, %d, %d ,%d, %d, %d, %d)",name,EnumToString(p
                           jaw_period,jaw_shift,teeth_period,teeth_shift,lips_period,l
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iGator 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iGator 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- GatorUpBuffer 배열이 iGator 지표의 심볼/주기 값보다 크면, 모든 것을 복사하지 않습
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- Gator Oscillator 지표 값 배열을 채웁니다
```

© 2000-2025, MetaQuotes Ltd.

```
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownColo
      shift,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Gator Oscillator 지표에서의 값 수 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iGator 지표에서 지표 버퍼 채우기                                 |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],          // 상단 히스토그램용 지표 버퍼
                           double &up_color_buffer[],     // 상단 히스토그램의 가격 지수에
                           double &downs_buffer[],        // 하단 히스토그램용 지표 버퍼
                           double &downs_color_buffer[], // 하위 히스토그램의 가격 지수에
                           int u_shift,                   // 상단 및 하단 히스토그램 시프트
                           int ind_handle,                // iGator 지표 핸들
                           int amount                     // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- GatorUpBuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iGator 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- GatorUpColors 배열 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iGator 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- GatorDownBuffer 배열의 일부를 인덱스 2가 있는 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("iGator 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
       return(false);
      }

//--- Gator DownColors 배열의 일부를 인덱스 3이 있는 지표 버퍼 값으로 채웁니다
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("iGator 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
       return(false);
      }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iIchimoku

이 함수는 Ichimoku Kinko Hyo 지표 핸들을 반환합니다.

```
int  iIchimoku(
   string           symbol,           // 심볼명
   ENUM_TIMEFRAMES  period,           // 주기
   int              tenkan_sen,       // 텐칸 센 주기
   int              kijun_sen,        // 기준 센 주기
   int              senkou_span_b     // Senkou Span B 주기
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

tenkan_sen

[in] 텐칸 센 평균 주기.

kijun_sen

[in] 기준 센 평균 주기.

senkou_span_b

[in] Senkou Span B 평균 주기.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 넘버: 0 - TENKANSEN_LINE, 1 - KIJUNSEN_LINE, 2 - SENKOUSPANA_LINE, 3 - SENKOUSPANB_LINE, 4 - CHIKOUSPAN_LINE.

예제:

```
//+------------------------------------------------------------------+
//|                                                Demo_iIchimoku.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iIchimoku 기술 지표의 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "표준적인 Ichimoku Kinko Hyo와 같은 다른 모든 파라미터."

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- the Tenkan_sen plot
#property indicator_label1  "Tenkan_sen"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- the Kijun_sen plot
#property indicator_label2  "Kijun_sen"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrBlue
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- the Senkou_Span plot
#property indicator_label3  "Senkou Span A;Senkou Span B" // 데이터 창에 두 필드가 표시됨
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- the Chikou_Span plot
#property indicator_label4  "Chinkou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,          // iIchimoku 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation             type=Call_iIchimoku;       // 함수 유형
input int                  tenkan_sen=9;              // 텐칸 센 주기
input int                  kijun_sen=26;              // 기준 센 주기
input int                  senkou_span_b=52;          // Senkou Span B 주기
input string               symbol=" ";                // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double         Tenkan_sen_Buffer[];
double         Kijun_sen_Buffer[];
```

© 2000-2025, MetaQuotes Ltd.

```
double          Senkou_Span_A_Buffer[];
double          Senkou_Span_B_Buffer[];
double          Chinkou_Span_Buffer[];
//--- ichimoku 지표 핸들을 저장하는 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- Ichimoku Kinko Hyo 지표 값의 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
   SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
   SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
   SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- 기준 센 막대의 센코우 스팬 채널의 시프트를 미래 방향으로 설정
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- Chinkou 데이터 스팬 때문에 Chikou Span 라인의 시프트를 설정할 필요가 없습니다
//--- ichimoku의 이동과 함께 이미 저장되어 있습니다
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[3];
      //--- 엘리게이터 선의 주기 및 이동
      pars[0].type=TYPE_INT;
      pars[0].integer_value=tenkan_sen;
      pars[1].type=TYPE_INT;
      pars[1].integer_value=kijun_sen;
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[2].type=TYPE_INT;
      pars[2].integer_value=senkou_span_b;
      //--- create handle
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 ilchimoku 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Ichimoku Kinko Hyo 지표가 계산되는 심볼/타임프레임을 표시
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- ilchimoku 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 계산 시작이거나 ilchimoku 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- Tenkan_sen_Buffer 배열이 심볼/주기에 대한 ichimoku 지표의 값 수보다 크면, 모든
       //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
       //--- 두 개 이하의 막대가 추가되었음을 의미합니다
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- Ichimoku Kinko Hyo 지표의 값으로 배열을 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,S
       kijun_sen,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Ichimoku Kinko Hyo 지표의 값수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| ichimoku 지표에서 지표 버퍼를 채웁니다                           |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // 텐칸 센 라인의 지표 버퍼
                           double &kijun_sen_buffer[],      // Kijun_sen 라인의 지표 버
                           double &senkou_span_A_buffer[],  // Senkou Span A 라인의 지표
                           double &senkou_span_B_buffer[],  // Senkou Span B 라인의 지표
                           double &chinkou_span_buffer[],   // Chinkou Span 라인의 지표
                           int senkou_span_shift,           // Senkou Span 라인의 미래
                           int ind_handle,                  // ichimoku 지표 핸들
                           int amount                       // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- Tenkan_sen_Buffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("1.지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
     }

//--- Kijun_sen_Buffer 배열의 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("2.ilchimoku 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- Chinkou_Span_Buffer 배열의 일부를 인덱스 2가 있는 지표 버퍼의 값으로 채웁니다
//--- senkou_span_shift>0인 경우 선은 senkou_span_shift 막대에 의해 미래 방향으로 이동됩니다
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("3.ilchimoku 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- Senkou_Span_A_Buffer 배열의 일부를 인덱스 3이 있는 지표 버퍼의 값으로 채웁니다
//--- senkou_span_shift>0인 경우 선은 senkou_span_shift 막대에 의해 미래 방향으로 이동됩니다
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("4.ilchimoku 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- Senkou_Span_B_Buffer 배열의 일부를 인덱스가 0인 지표 버퍼 값으로 채웁니다
//--- Chinkou Span 데이터를 복사 할 때, Chinkou Span 데이터 때문에 이동을 고려할 필요가 없습니
//--- ilchimoku의 이동과 함께 이미 저장되어 있습니다
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("5.ilchimoku 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iBWMFI

이 함수는 시장 촉진 지수 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iBWMFI(
   string               symbol,              // 심볼명
   ENUM_TIMEFRAMES      period,              // 주기
   ENUM_APPLIED_VOLUME  applied_volume       // 계산을 위한 볼륨 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

applied_volume

[in] 사용한 볼륨. ENUM_APPLIED_VOLUME 상수 중 하나일 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iBWMFI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iBWMFI 기술 지표에 대한 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- the iBWMFI plot
#property indicator_label1  "iBWMFI"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrLime,clrSaddleBrown,clrBlue,clrPink
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,             // iBWMFI 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iBWMFI;          // 함수 유형
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;// 볼륨 유형
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          iBWMFIBuffer[];
double          iBWMFIColors[];
//--- iBWMFI 지표 핸들을 저장하기 위한 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 빌 윌리엄스 지표에 의한 시장 촉진 지수의 값 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iBWMFI)
      handle=iBWMFI(name,period,applied_volume);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- 볼륨 유형
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iBWMFI 지표 핸들 생성 실패, 에러 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Bill Williams 지표가 계산된 에 대한 기호/타임프레임을 보여줍니다
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iBWMFI 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표 계산을 처음 시작하는 경우 또는 iBWMFI 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
```

© 2000-2025, MetaQuotes Ltd.

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iBWMFIBuffer 배열이 심볼/주기에 대한 iBWMFI 지표 값 수보다 크면 모든 항목을 복사하
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- Bill Williams의 시장 촉진 지수 지표 값으로 배열을 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 빌 윌리엄스 지표에 의한 시장 촉진 지수의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBWMFI 지표에서 지표 버퍼 채우기                                 |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],     // 히스토그램 값에 대한 지표 버퍼
                           double &colors[],     // 히스토그램 색상의 지표 버퍼
                           int ind_handle,       // iBWMFI 지표 핸들
                           int amount            // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iBWMFI 버퍼 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iBWMFI 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- iBWMFIColors 배열의 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("iBWMFI 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
       return(false);
      }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMomentum

이 함수는 모멘텀 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iMomentum(
   string                symbol,            // 심볼명
   ENUM_TIMEFRAMES       period,            // 주기
   int                   mom_period,        // 평균 기간
   ENUM_APPLIED_PRICE    applied_price      // 가격 유형 또는 핸들
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

mom_period

[in] 가격 변동 계산을 위한 평균 기간(막대 수).

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                Demo_iMomentum.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iMomentum 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 표준 모멘텀과 유사합니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iMomentum
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,          // iMomentum 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iMomentum;       // 함수 유형
input int                   mom_period=14;             // Momentum 기간
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 가격 유형
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          iMomentumBuffer[];
//--- iMomentum 지표 핸들 저장 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 모멘텀 지표 값의 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iMomentum)
```

© 2000-2025, MetaQuotes Ltd.

```
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[2];
      //--- period
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- 가격 유형
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iMomentum 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Momentum 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iMomentum 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iMomentum 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iMomentumBuffer 배열이 심볼/주기에 대한 iMomentum 지표 값 수보다 크면, 모든 것을
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMomentumBuffer 배열에 Momentum 지표 값을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Momentum 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMomentum 지표에서 지표 버퍼 채우기                              |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Momentum 값의 지표 버퍼
                         int ind_handle,    // iMomentum 지표 핸들
                         int amount         // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iMomentumBuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
```

© 2000-2025, MetaQuotes Ltd.

```
      PrintFormat("지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMFI

이 함수는 화폐 흐름 지수 지표 핸들을 반환합니다.

```
int  iMFI(
   string               symbol,              // 심볼명
   ENUM_TIMEFRAMES      period,              // 주기
   int                  ma_period,           // 평균화 기간
   ENUM_APPLIED_VOLUME  applied_volume       // 계산을 위한 볼륨 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 계산을 위한 평균 기간(바 카운트).

applied_volume

[in] 사용한 볼륨. ENUM_APPLIED_VOLUME 값 중 하나일 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMFI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iMFI 기술 지표용 지표 버퍼 크기."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 매개 변수는 표준 자금 흐름 지수와 유사합니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iMFI plot
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- horizontal levels in the indicator window
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // iMFI 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iMFI;              // 함수 유형
input int                   ma_period=14;                // 기간
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;  // 볼륨 유형
input string                symbol=" ";                  // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // 타임프레임
//--- 지표 버퍼
double          iMFIBuffer[];
//--- iMFI 지표 핸들을 보관하기 위한 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 자금 흐름 지수 지표에 있는 값의 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 지표 핸들 생성
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[2];
      //--- period
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 볼륨 유형
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iMFI 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 자금 흐름 지수 지표가 계산되는 심볼/타임테이블을 표시합니다
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iMFI 지표에서 복사된 값 수
```

© 2000-2025, MetaQuotes Ltd.

```
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 계산 시작이거나 iMFI 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iMFIBuffer 배열이 심볼/주기 iMFI 지표 값 수보다 크면 모든 것을 복사하지 않습니다
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMFIBuffer 배열에 자금흐름지수 지표 값을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 자금 흐름 지수 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMFI 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // 자금 흐름 지수 값의 지표 버퍼
                         int ind_handle,    // iMFI 지표 핸들
                         int amount         // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iMFIBuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iMFI 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMA

이 함수는 이동 평균 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iMA(
   string                symbol,              // 심볼명
   ENUM_TIMEFRAMES       period,              // 주기
   int                   ma_period,           // 평균 주기
   int                   ma_shift,            // 수평 이동
   ENUM_MA_METHOD        ma_method,           // 평활 유형
   ENUM_APPLIED_PRICE    applied_price        // 가격 유형 또는 핸들
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 이동 평균 계산을 위한 평균 기간.

ma_shift

[in] 가격 차트에 상대적인 지표 이동.

ma_method

[in] 평활 유형. ENUM_MA_METHOD 값 중 하나일 수 있습니다.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                    Demo_iMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iMA 기술 지표용 지표 버퍼의 크기."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "표준 이동 평균과 같은 다른 모든 매개 변수."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iMA plot
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,               // use iMA
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation             type=Call_iMA;               // 함수 유형
input int                  ma_period=10;                // ma 주기
input int                  ma_shift=0;                  // 시프트
input ENUM_MA_METHOD       ma_method=MODE_SMA;          // 평활 유형
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;   // 가격 유형
input string               symbol=" ";                  // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // 타임프레임
//--- 지표 버퍼
double         iMABuffer[];
//--- iMA 지표 핸들을 저장하는 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 이동 평균 지표 값의 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
//--- 시프트 세팅
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 지표가 그려지는 심볼 결정
```

© 2000-2025, MetaQuotes Ltd.

```
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- 구조물을 지표의 매개변수로 채움
      MqlParam pars[4];
      //--- period
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 이동
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 평활유형
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- 가격 유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iMA 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 이동 평균 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period, ma_shift,EnumToString(ma_method),EnumToString(ap
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iMA 지표로부터 복제된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iMA 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iMABuffer 배열이 심볼/주기에 대한 iMA 지표의 값 수보다 크면, 모든 것을 복사하지 않
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMABuffer 배열에 이동 평균 지표 값을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 이동 평균 지표의 값 수를 기억
```

© 2000-2025, MetaQuotes Ltd.

```
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| MA 지표에서 지표 버퍼 채우기                                     |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // 이동 평균 값의 지표 버퍼
                         int shift,          // 시프트
                         int ind_handle,     // iMA 지표 핸들
                         int amount          // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iMABuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iMA 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iOsMA

이 함수는 오실레이터 지표의 이동 평균 핸들을 반환합니다. OsMA 오실레이터는 MACD 값과 신호 라인 간의 차이를 표시합니다. 버퍼가 하나뿐입니다.

```
int  iOsMA(
   string              symbol,              // 심볼명
   ENUM_TIMEFRAMES     period,              // 주기
   int                 fast_ema_period,     // 빠른 이동 평균 기간
   int                 slow_ema_period,     // 느린 이동 평균 기간
   int                 signal_period,       // 그들의 차이에 대한 평균 기간
   ENUM_APPLIED_PRICE  applied_price        // 가격 유형 또는 핸들
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

fast_ema_period

[in] 빠른 이동 평균 계산 기간.

slow_ema_period

[in] 느린 이동 평균 계산 기간.

signal_period

[in] 신호 라인 계산을 위한 평균 주기.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

일부 시스템에서는 이 오실레이터를 MACD 히스토그램이라고도 합니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOsMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iOsMA 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 진동기의 표준 이동 평균과 유사합니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iOsMA plot
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,             // iOsMA 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iOsMA;           // 함수 유형
input int                   fast_ema_period=12;        // 빠른 ma 기간
input int                   slow_ema_period=26;        // 느린 ma 기간
input int                   signal_period=9;           // 차이의 평균화 기간
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 가격 유형
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          iOsMABuffer[];
//--- iAMA 지표 핸들을 저장하기 위한 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 이동 평균 지표 값의 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
```

© 2000-2025, MetaQuotes Ltd.

```
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[4];
      //--- 빠른 ma 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- 느린 ma 주기
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- 빠른 이동 평균과 느린 이동 평균의 차이 평균 기간
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- 가격 유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 페어용 iOsMA 핸들 생성 실패, 오류 코드는 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 오실레이터 지표의 이동 평균이 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
```

© 2000-2025, MetaQuotes Ltd.

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iOsMA 지표로부터 복제된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표 첫 계산 시작이거나 iOsMA 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iOsMABuffer 배열이 심볼/주기에 대한 iOsMA 지표 값 수보다 크면, 모든 것을 복사하지
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iOsMA 지표 값으로 배열 채우기
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment(comm);
//--- 오실레이터 지표의 이동 평균 값 수 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iOsMA 지표에서 지표 버퍼 채우기                                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // OsMA 값 지표 버퍼
                         int ind_handle,        // iOsMA 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iOsMABuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iOsMA 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iMACD

이 함수는 이동 평균 수렴/발산 지표 핸들을 반환합니다. OsMA를 MACD 히스토그램이라고 하는 시스템에서는 이 지표가 두 줄로 표시됩니다. 클라이언트 터미널에서 이동 평균 수렴/발산은 히스토그램처럼 보입니다.

```
int  iMACD(
   string              symbol,              // 심볼명
   ENUM_TIMEFRAMES     period,              // 주기
   int                 fast_ema_period,     // 빠른 평균 계산을 위한 기간
   int                 slow_ema_period,     // 느린 평균 계산을 위한 기간
   int                 signal_period,       // 차이 평균화를 위한 기간
   ENUM_APPLIED_PRICE  applied_price        // 가격 유형 또는 핸들
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

fast_ema_period

[in] 빠른 이동 평균 계산 기간.

slow_ema_period

[in] 느린 이동 평균 계산 기간.

signal_period

[in] 시그널 라인 계산 기간.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 넘버는 다음과 같습니다: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMACD.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
```

© 2000-2025, MetaQuotes Ltd.

```
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iMACD 기술 지표용 지표 버퍼의 경우."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "표준 MACD와 같은 다른 모든 매개 변수."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- the MACD plot
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- the Signal plot
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,              // iMACD 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation             type=Call_iMACD;           // 함수 유형
input int                  fast_ema_period=12;        // 빠른 ma 기간
input int                  slow_ema_period=26;        // 느린 ma 기간
input int                  signal_period=9;           // 차이의 평균화 기간
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE; // 가격 유형
input string               symbol=" ";                // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double         MACDBuffer[];
double         SignalBuffer[];
//--- iMACD 지표 핸들 저장 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 이동 평균 수렴/분산 지표의 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[4];
      //--- 빠른 ma 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- 느린 ma 주기
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- 빠른 이동 평균과 느린 이동 평균의 차이 평균 기간
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- 가격 유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iMACD 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
```

© 2000-2025, MetaQuotes Ltd.

```
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 이동 평균 수렴/분산 지표가 계산되는 기호/시간 프레임 표시
   short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                           fast_ema_period,slow_ema_period,signal_period,EnumToString
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iMACD 지표에서 복사한 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iMACD 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- 지표의 첫 번째 계산 시작이거나 iMACD 지표 값 수가 변경된 경우
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 배열에 iMACD 지표 값을 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 이동 평균 지표 수렴/분산 값의 수 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMACD 지표에서 지표 버퍼 채우기                                  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],    // MACD 값 지표 버퍼
                           double &signal_buffer[],  // MACD 시그널 선 지표 버퍼
                           int ind_handle,           // iMACD 지표 핸들
                           int amount                // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iMACDBuffer 배열 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iMACD 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }

//--- 인덱스 1이 있는 지표 버퍼의 값으로 SignalBuffer 배열의 일부를 채웁니다
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iMACD 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iOBV

이 함수는 거래량 균형(On Balance Volume) 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iOBV(
   string                symbol,             // 심볼명
   ENUM_TIMEFRAMES       period,             // 기간
   ENUM_APPLIED_VOLUME   applied_volume      // 계산을 위한 볼륨 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

applied_volume

[in] 사용한 볼륨. ENUM_APPLIED_VOLUME 값 중 하나일 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOBV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iOBV 기술 지표용 지표 버퍼의 크기."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---  iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,             // iOBV 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iOBV;               // 함수 유형
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // 볼륨 유형
input string                symbol=" ";                   // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 타임프레임
//--- 지표 버퍼
double          iOBVBuffer[];
//--- iOBV 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 거래량 균형(On Balance Volume) 지표 값 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[1];
      //--- 볼륨 유형
```

© 2000-2025, MetaQuotes Ltd.

```
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼을 위한 iOBV 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 거래량 균형(On Balance Volume) 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iOBV 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 이것이 이 지표의 첫 계산 시작이거나 iOBV 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- iOBVBuffer 배열이 심볼/주기에 대한 iOBV 지표 값 수보다 크면 모든 것을 복사하지 않
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열에 iOBV 지표 값을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- On Balance Volume 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iOBV 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // OBV 값 지표 버퍼
                         int ind_handle,        // iOBV 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iOBVBuffer 배열 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iOBV 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iSAR

이 함수는 파라볼릭 SAR 시스템 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iSAR(
   string           symbol,      // 심볼명
   ENUM_TIMEFRAMES  period,      // 기간
   double           step,        // 가격 상승 단계 - 가속 계수
   double           maximum      // 스텝의 최대값
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

step

[in] 가격 상승의 단계, 보통 0.02입니다.

maximum

[in] 최대 단계, 보통 0.2입니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iSAR.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iSAR 기술 지표용 지표 버퍼의 크기."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 표준 파라볼릭 SAR과 유사합니다."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- drawing iSAR
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,              // iSAR 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iSAR;              // 함수 유형
input double                step=0.02;                   // 단계 - 정지 후행 가속 계수
input double                maximum=0.2;                 // 스텝의 최대값
input string                symbol=" ";                  // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // 타임프레임
//--- 지표 버퍼
double          iSARBuffer[];
//--- iSAR 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 파라볼릭 SAR 지표에서 값의 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- 차트에 표시하기 위한 PLOT_ARROW 속성에 대한 Wingdings charset에서 기호 코드를 설정합니다
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 지표 핸들 생성
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[2];
      //--- step value
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- 계산에 사용할 수 있는 단계 값의 한계
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iSAR 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 파라볼릭 SAR 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iSAR 지표에서 복사된 값 수
   int values_to_copy;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iSAR 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iSARBuffer 배열이 심볼/주기에 대한 iSAR 지표 값 수보다 크면, 모든 것을 복사하지 않
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열에 iSAR 지표 값을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 파라볼릭 SAR 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iSAR 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],  // 파라볼릭 SAR 값의 지표 버퍼
                         int ind_handle,        // iSAR 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iSARBuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("i SAR 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
       return(false);
      }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iRSI

이 함수는 상대 강도 지수 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iRSI(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  ma_period,        // 평균 주기
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] RSI 계산을 위한 평균 주기.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRSI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iRSI 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 매개변수는 표준 상대 강도 지수와 유사합니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- drawing iRSI
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- limits for displaying of values in the indicator window
#property indicator_maximum 100
#property indicator_minimum 0
//--- horizontal levels in the indicator window
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // iRSI 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iRSI;               // 함수 유형
input int                   ma_period=14;                 // 평균 주기
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // 가격 유형
input string                symbol=" ";                   // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 타임프레임
//--- 지표 버퍼
double          iRSIBuffer[];
//--- iRSI 지표 핸들을 저장하는 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 상대 강도 지수 지표의 값의 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 지표가 부착된 차트의 심볼을 취합니다
       name=_Symbol;
      }
//--- 지표 핸들 생성
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
      {
       //--- 지표 매개변수로 구조를 채웁니다
       MqlParam pars[2];
       //--- 이동 평균 주기
       pars[0].type=TYPE_INT;
       pars[0].integer_value=ma_period;
       //--- 계산에 사용할 수 있는 단계 값의 한계
       pars[1].type=TYPE_INT;
       pars[1].integer_value=applied_price;
       handle=IndicatorCreate(name,period,IND_RSI,2,pars);
      }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
      {
       //--- 실패에 대해 구분하고 오류 코드를 출력
       PrintFormat("%s/%s 심볼에 대한 iRSI 지표 핸들 생성 실패, 오류 코드 %d",
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- 지표가 일찍 중단됨
       return(INIT_FAILED);
      }
//--- 상대 강도 지수 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const int &spread[])
  {
//--- iRSI 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iRSI 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iRSIBuffer 배열이 심볼/주기용 iRSI 지표 값 수보다 크면 모든 것을 복사하지 않습니다
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열에 iRSI 지표 값을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 상대 강도 지수 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iRSI 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rsi_buffer[],  // 상대 강도 지수 값의 지표 버퍼
                         int ind_handle,        // iRSI 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 오류 코드 재설정
   ResetLastError();
//--- iRSIBuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iRSI 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iRVI

이 함수는 상대 활력 지수 지표 핸들을 반환합니다.

```
int  iRVI(
   string           symbol,        // 심볼명
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균 주기
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] RVI 계산용 평균 기간

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 넘버는 다음과 같습니다: 0 -MAIN_LINE, 1 -SIGNAL_LINE.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRVI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iRVI 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 표준의 상대적 활력 지수와 유사합니다."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- the RVI plot
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- the Signal plot
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // iRVI 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation             type=Call_iRVI;              // 함수 유형
input int                  ma_period=10;                // 계산 주기
input string               symbol=" ";                  // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // 타임프레임
//--- 지표 버퍼
double         RVIBuffer[];
double         SignalBuffer[];
//--- iRVI 지표 핸들을 저장하는 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 상대 활력 지수 지표 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
```

© 2000-2025, MetaQuotes Ltd.

```
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[1];
      //--- 계산 기간
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iRVI 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 상대 활력 지수 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iRVI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iRVI 지표에서 복사된 값 수
```

© 2000-2025, MetaQuotes Ltd.

```
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iRVI 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- RVIBuffer 배열이 심볼/주기에 대한 iRVI 지표 값 수보다 크면 모든 것을 복사하지 않습
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iRVI 지표 값으로 배열 채우기
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 상대 활력 지수 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iRVI 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // 상대 활력 지수 값의 지표 버퍼
                         double &signal_buffer[],  // 시그널 라인의 지표 버퍼
                         int ind_handle,           // iRVI 지표 핸들
                         int amount                // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
```

© 2000-2025, MetaQuotes Ltd.

```
//--- iRVI Buffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iRVI 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 인덱스 1이 있는 지표 버퍼의 값으로 Signal Buffer 배열의 일부를 채웁니다
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iRVI 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iStdDev

이 함수는 표준 편차 지표의 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iStdDev(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  ma_period,        // 평균 주기
   int                  ma_shift,         // 수평 이동
   ENUM_MA_METHOD       ma_method,        // 평활 유형
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 지표 계산의 평균 기간.

ma_shift

[in] 가격 차트에 상대적인 지표 이동.

ma_method

[in] 평균화 유형. ENUM_MA_METHOD 값 중 하나일 수 있습니다.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                Demo_iStdDev.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iStdDev 기술 지표의 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 정규 표준 편차와 유사합니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iStdDev plot
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,            // iStdDev 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iStdDev;          // 함수 유형
input int                   ma_period=20;               // 평균화 기간
input int                   ma_shift=0;                 // 이동
input ENUM_MA_METHOD        ma_method=MODE_SMA;         // 평활 유형
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 가격 유형
input string                symbol=" ";                 // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;      // 타임프레임
//--- 지표 버퍼
double          iStdDevBuffer[];
//--- iStdDev 지표 핸들을 저장하는 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 표준 편차 지표 값의 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
//--- 시프트 세팅
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 지표가 그려지는 심볼 결정
```

© 2000-2025, MetaQuotes Ltd.

```
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[4];
      //--- period
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 이동
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 평활유형
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- 가격 유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼용 iStdDev 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 표준 편차 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(ma_method),EnumToString(app
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iStdDev 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iStdDev 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iStdDevBuffer 배열이 심볼/기간 동안 iStdDev 지표 값 수보다 크면, 모든 것을 복사하
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 배열을 표준 편차 지표 값으로 채움
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 표준 편차 지표 값 수를 기억
```

© 2000-2025, MetaQuotes Ltd.

```
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iStdDev 지표에서 지표 버퍼 채우기                                |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // 표준 편차 선의 지표 버퍼
                         int std_shift,         // 표준 편차 선의 이동
                         int ind_handle,        // iStdDev 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iStdDevBuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iStdDev 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iStochastic

이 함수는 확률적 오실레이터 지표 핸들을 반환합니다.

```
int  iStochastic(
   string           symbol,          // 심볼명
   ENUM_TIMEFRAMES  period,          // 주기
   int              Kperiod,         // K-주기 (계산을 위한 막대 수)
   int              Dperiod,         // D-주기 (첫 평활 주기)
   int              slowing,         // 최종 평활
   ENUM_MA_METHOD   ma_method,       // 평활 유형
   ENUM_STO_PRICE   price_field      // 확률적 연산 방법
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

Kperiod

[in] %K 라인 계산을 위한 평균 기간(바 카운트).

Dperiod

[in] %D 라인 계산을 위한 평균 기간(바 카운트).

slowing

[in] 둔화 값.

ma_method

[in] 평균화 유형. ENUM_MA_METHOD 값 중 하나일 수 있습니다.

price_field

[in] 계산을 위한 가격 선택 매개변수. ENUM_STO_PRICE 값 중 하나일 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

참고

버퍼 넘버: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

예제:

```
//+------------------------------------------------------------------+
//|                                              Demo_iStochastic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iStochastic 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 파라미터는 표준 확률 오실레이터와 유사합니다."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- the Stochastic plot
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- the Signal plot
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- set limit of the indicator values
#property indicator_minimum 0
#property indicator_maximum 100
//--- horizontal levels in the indicator window
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,       // iStochastic 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation             type=Call_iStochastic;     // 함수 유형
input int                  Kperiod=5;                 // K 주기 (계산용 막대의 수)
input int                  Dperiod=3;                 // D 주기 (일차 평활 기간)
input int                  slowing=3;                 // 최종 평활 기간
input ENUM_MA_METHOD       ma_method=MODE_SMA;        // 평활 유형
input ENUM_STO_PRICE       price_field=STO_LOWHIGH;   // 확률적 계산법
input string               symbol=" ";                // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // 타임프레임
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 지표 버퍼
double         StochasticBuffer[];
double         SignalBuffer[];
//--- iStochastic 지표 핸들을 저장하기 위한 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 확률적 오실레이터 지표 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[5];
      //--- 연산용 K 주기
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- 첫 평활용 D 주기
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- 첫 평활용 K 주기
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
      //--- 평활유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ma_method;
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 확률적 계산법
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼에 대한 iStochastic 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 확률적 오실레이터 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %s)",name,EnumToString
                           Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iStochastic 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iStochastic 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- StochasticBuffer 배열이 심볼/주기에 대한 iStochastic 지표의 값 수보다 크면, 모든
       //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
       //--- 두 개 이하의 막대가 추가되었음을 의미합니다
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- 배열을 iStochastic 지표 값으로 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) ret
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 확률적 오실레이터 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iStochastic 지표의 지표 버퍼 채우기                              |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // 확률적 오실레이터 값의 지표 버퍼
                           double &signal_buffer[],  // 시그널 라인의 지표 버퍼
                           int ind_handle,           // iStochastic 지표 핸들
                           int amount                // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- StochasticBuffer 배열의 일부를 인덱스가 0인 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("iStochastic 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
       //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
       return(false);
      }
//--- 인덱스 1이 있는 지표 버퍼의 값으로 SignalBuffer 배열의 일부를 채웁니다
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
      {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iStochastic 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iTEMA

이 함수는 삼중 지수 이동 평균 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iTEMA(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  ma_period,        // 평균 주기
   int                  ma_shift,         // 지표의 수평 이동
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 계산을 위한 평균 기간(바 카운트).

ma_shift

[in] 가격 차트에 상대적인 지표 이동.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iTEMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iTEMA 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "다른 모든 매개변수는 표준 삼중 지수 이동 평균과 비슷합니다."
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iTEMA plot
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,             // iTEMA 사용
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iTEMA;           // 함수 유형
input int                   ma_period=14;              // 평균화 기간
input int                   ma_shift=0;                // 이동
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 가격 유형
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          iTEMABuffer[];
//--- iTEMA 지표 핸들을 저장하는 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 삼중 지수 이동 평균 지표 값의 수를 유지할 것입니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- 시프트 세팅
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
```

© 2000-2025, MetaQuotes Ltd.

```
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[3];
      //--- period
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 이동
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- 가격 유형
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼용 iTEMA 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 삼중 지수 이동 평균 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
```

© 2000-2025, MetaQuotes Ltd.

```
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iTEMA 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iTEMA 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iTEMABuffer 배열이 기호/주기에 대한 iTEMA 지표 값 수보다 크면 모든 것을 복사하지
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 삼중 지수 이동 평균 지표 값으로 배열을 채웁니다
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 삼중 지수 이동 평균 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iTEMA 지표에서 지표 버퍼 채우기                                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // 삼중 지수 이동 평균 값의 지표 버퍼
```

© 2000-2025, MetaQuotes Ltd.

```
                           int t_shift,            // 선 이동
                           int ind_handle,         // iTEMA 지표 핸들
                           int amount              // 복제된 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iTEMABuffer 배열의 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채우기
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iTEMA 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iTriX

이 함수는 삼중 지수 이동 평균 오실레이터 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iTriX(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  ma_period,        // 평균 주기
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

ma_period

[in] 계산을 위한 평균 기간(바 카운트).

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iTriX.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iTriX 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iTriX plot
#property indicator_label1  "iTriX"
```

© 2000-2025, MetaQuotes Ltd.

```
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,              // iTriX 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iTriX;           // 함수 유형
input int                   ma_period=14;              // 기간
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 가격 유형
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          iTriXBuffer[];
//--- iTriX 지표 핸들을 저장하는 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 삼중 지수 이동 평균 오실레이터의 값 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iTriX)
      handle=iTriX(name,period,ma_period,applied_price);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[2];
      //--- period
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- 가격 유형
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼용 iTriX 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 삼중 지수 이동 평균 오실레이터가 계산한 심볼/타임프레임 표시
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iTriX 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
       return(0);
      }
//--- 지표의 첫 번째 계산 시작이거나 iTriX 지표의 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
      {
       //--- iTriXBuffer 배열이 심볼/주기에 대한 iTriX 지표 값 수보다 크면, 모든 것을 복사하지
       //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
   else
      {
       //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
       //--- 두 개 이하의 막대가 추가되었음을 의미합니다
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- 삼중 지수 이동 평균 오실레이터의 값으로 배열 채우기
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 삼중 지수 이동 평균 오실레이터의 값 수 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iTriX 지표에서 지표 버퍼 채우기                                  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // 삼중 지수 이동 평균 오실레이터 값의 지표
                         int ind_handle,        // iTriX 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iTriXBuffer 배열 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
      {
       //--- 복사가 실패하면 오류 코드를 알려줍니다
       PrintFormat("iTriX 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iWPR

이 함수는 Larry Williams의 백분율 범위 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iWPR(
   string           symbol,          // 심볼명
   ENUM_TIMEFRAMES  period,          // 주기
   int              calc_period      // 평균화 기간
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

calc_period

[in] 지표 계산을 위한 주기(바 카운트).

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iWPR.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iWPR 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iWPR plot
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrCyan
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//--- set limit of the indicator values
#property indicator_minimum -100
#property indicator_maximum 0
//--- horizontal levels in the indicator window
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // 사용 iWPR
   Call_IndicatorCreate    // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation             type=Call_iWPR;           // 함수 유형
input int                  calc_period=14;           // 기간
input string               symbol=" ";               // 심볼
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;    // 타임프레임
//--- 지표 버퍼
double         iWPRBuffer[];
//--- iWPR 지표 핸들 저장 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 래리 윌리엄스의 백분율 범위 지표에 있는 값의 수를 유지할 것입니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iWPR)
```

© 2000-2025, MetaQuotes Ltd.

```
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[1];
      //--- period
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼용 iWPR 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 윌리엄스의 백분율 범위 지표가 계산될 심볼/타임프레임 표시
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iWPR 지표에서 복사된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 지표 첫 계산 시작이거나 iWPR 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iWPRBuffer 배열이 기호/주기에 대한 iWPR 지표 값 수보다 크면 모든 것을 복사하지 않
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 윌리엄스 백분율 범위 지표 값으로 배열 채우기
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 래리 윌리엄스의 백분율 범위 지표 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iWPR 지표에서 지표 버퍼 채우기                                   |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[],  // 윌리엄스 백분율 범위 값의 지표 버퍼
                         int ind_handle,        // iWPR 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- iWPRBuffer 배열 일부를 0 인덱스를 가진 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iWPR 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iVIDyA

이 함수는 가변 지수 동적 평균 지표 핸들을 반환합니다. 버퍼가 하나뿐입니다.

```
int  iVIDyA(
   string               symbol,           // 심볼명
   ENUM_TIMEFRAMES      period,           // 기간
   int                  cmo_period,       // 샹드 모멘텀 주기
   int                  ema_period,       // EMA 평활 기간
   int                  ma_shift,         // 가격 차트에서 수평 이동
   ENUM_APPLIED_PRICE   applied_price     // 가격 또는 핸들 유형
   );
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

cmo_period

[in] 샹드 모멘텀 오실레이터 연산용 기간 (바 카운트).

ema_period

[in] 평활 요인 연산용 EMA 기간 (바 카운트).

ma_shift

[in] 가격 차트에 상대적인 지표 이동.

applied_price

[in] 사용된 가격. 가격 상수 ENUM_APPLIED_PRICE 또는 다른 지표의 핸들이 될 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iVIDyA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iVIDyA 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
```

© 2000-2025, MetaQuotes Ltd.

```
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."
#property description "표준 가변 지수 동적 평균과 같은 다른 모든 매개 변수."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- the iVIDyA plot
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,             // iVIDyA 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iVIDyA;          // 함수 유형
input int                   cmo_period=15;             // 샹드 모멘텀 기간
input int                   ema_period=12;             // 평활 요인 기간
input int                   ma_shift=0;                // 이동
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // 가격 유형
input string                symbol=" ";                // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // 타임프레임
//--- 지표 버퍼
double          iVIDyABuffer[];
//--- iVIDyA 지표 핸들 저장 변수
int    handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 가변 지수 동적 평균 지표에서의 값의 수를 유지합니다
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
//--- 시프트 세팅
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- 지표가 그려지는 심볼 결정
```

© 2000-2025, MetaQuotes Ltd.

```
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
      MqlParam pars[4];
      //--- 샹드 모멘텀 기간
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- 평활 요인 기간
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- 이동
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- 가격 유형
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼용 iVIDyA 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- 가변 지수 동적 평균 지표가 계산되는 심볼/타임프레임 표시
   short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %s)",name,EnumToString(period),
                           cmo_period,ema_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iVIDyA로부터 복제된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표의 첫 번째 계산 시작이거나 iVIDyA 지표 값 수가 변경된 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- iVIDyABuffer 배열이 심볼/주기에 대한 iVIDyA 지표 값 수보다 큰 경우, 모든 것을 복
      //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
      //--- 두 개 이하의 막대가 추가되었음을 의미합니다
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- 가변 지수 동적 평균 지표 값으로 배열 채우기
//--- FillArrayFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- 가변 지수 동적 평균 지표 값 수를 기억
```

© 2000-2025, MetaQuotes Ltd.

```
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iVIDyA 지표에서 지표 버퍼 채우기                                 |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// 가변 지수 동적 평균 값의 지표 버퍼
                         int v_shift,           // 라인 이동
                         int ind_handle,        // iVIDyA 지표 핸들
                         int amount             // 복제된 값 수
                         )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- 가변 지수 동적 평균 값의 지표 버퍼
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iVIDyA 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

iVolumes

이 함수는 볼륨 지표 핸들을 반환합니다. 이것은 한 개의 버퍼만 가지고 있습니다.

```
int  iVolumes(
   string                symbol,              // 심볼명
   ENUM_TIMEFRAMES       period,              // 주기
   ENUM_APPLIED_VOLUME   applied_volume       // 계산을 위한 볼륨 유형
   )
```

Parameters

symbol

[in] 지표를 계산하는 데 사용해야 하는 보안의 기호 이름. NULL 값은 현재 심볼을 의미합니다.

period

[in] 기간의 값은 ENUM_TIMEFRAMES 값 중 하나일 수 있으며, 0은 현재 기간을 의미합니다.

applied_volume

[in] 이용된 볼륨. ENUM_APPLIED_VOLUME 값 중 하나일 수 있습니다.

반환 값

오류가 INVALID_HANDLE를 반환하는 경우 지정된 기술 지표의 핸들을 반환합니다. 컴퓨터 메모리는 지표 핸들이 전달되는 IndicatorRelease() 함수를 사용하여 더 이상 사용되지 않는 지표에서 해방될 수 있습니다.

예제:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iVolumes.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "이 지표는 다음 데이터를 얻는 방법을 보여줍니다:"
#property description "iVolumes 기술 지표용 지표 버퍼."
#property description "지표 계산에 사용되는 심볼 및 타임프레임은"
#property description "심볼 및 주기 매개변수로 설정됩니다."
#property description "핸들을 만드는 방법은 'type' 매개변수(함수 유형)를 통해 설정됩니다."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- the iVolumes plot
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 핸들 생성 방법 열거                                              |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,           // iVolumes 사용
   Call_IndicatorCreate     // IndicatorCreate 사용
  };
//--- 입력 매개변수
input Creation              type=Call_iVolumes;           // 함수 유형
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // 볼륨 유형
input string                symbol=" ";                   // 심볼
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // 타임프레임
//--- 지표 버퍼
double          iVolumesBuffer[];
double          iVolumesColors[];
//--- iVolumes 지표 핸들을 저장하는 변수
int     handle;
//--- 저장 변수
string name=symbol;
//--- 차트의 지표명
string short_name;
//--- 볼륨 지표의 값 수를 유지합니다
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| 커스텀 지표 초기화 함수                                          |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 지표 버퍼에 배열 할당
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- 지표가 그려지는 심볼 결정
   name=symbol;
//--- 오른쪽과 왼쪽의 공백을 삭제
   StringTrimRight(name);
   StringTrimLeft(name);
//--- 'name' 문자열의 길이가 0이 되는 경우
   if(StringLen(name)==0)
     {
      //--- 지표가 부착된 차트의 심볼을 취합니다
      name=_Symbol;
     }
//--- 지표 핸들 생성
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
   else
     {
      //--- 지표 매개변수로 구조를 채웁니다
```

© 2000-2025, MetaQuotes Ltd.

```
      MqlParam pars[1];
      //--- 가격 유형
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
     }
//--- 핸들이 생성되지 않은 경우
   if(handle==INVALID_HANDLE)
     {
      //--- 실패에 대해 구분하고 오류 코드를 출력
      PrintFormat("%s/%s 심볼용 iVolumes 지표 핸들 생성 실패, 오류 코드 %d",
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- 지표가 일찍 중단됨
      return(INIT_FAILED);
     }
//--- Volumes 지표가 계산될 심볼/타임프레임 표기
   short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- 지표의 정상 초기화
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 커스텀 지표 반복 함수                                            |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
//--- iVolumes 지표로부터 복제된 값 수
   int values_to_copy;
//--- 지표에서 계산된 값 수를 결정
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() 가 %d 을(를) 반환, 오류 코드 %d",calculated,GetLastE
      return(0);
     }
//--- 지표 계산의 첫 시작일 경우 또는 iVolumes 지표 값 수가 변화될 경우
//---또는 두 개 이상의 막대에 대한 지표 계산이 필요한지 여부(가격 이력에서 무언가가 변경되었음을
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- iVolumesBuffer 배열이 심볼/기간용 iVolumes 지표의 값 수보다 크다면, 아무것도 복사
       //--- 모든 항목을 복사하지 않고 지표ㅍ 버퍼 크기보다 작게 복사합니다.
       if(calculated>rates_total) values_to_copy=rates_total;
       else                       values_to_copy=calculated;
      }
    else
      {
       //--- 이는 지표 계산의 첫 번째가 아니며 계산을 위한 OnCalculate()의 마지막 호출이
       //--- 두 개 이하의 막대가 추가되었음을 의미합니다
       values_to_copy=(rates_total-prev_calculated)+1;
      }
//--- iVolumes 지표 값의 배열을 채웁니다
//--- FillArraysFromBuffer가 false를 반환하면 정보가 아직 준비되지 않았음을 의미하므로 작업을
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) ret
//--- 메시지 형성
   string comm=StringFormat("%s ==>  지표 %s 값 업데이트: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- 차트에 서비스 메시지 표시
   Comment(comm);
//--- Volumes 지표의 값 수를 기억
   bars_calculated=calculated;
//--- 다음 호출에 대해 prev_calculated 값을 반환합니다
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iVolumes 지표로부터 지표 버퍼 채우기                             |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // 볼륨 값의 지표 버퍼
                           double &color_buffer[],    // 색상의 지표 버퍼
                           int ind_handle,            // iVolumes 지표 핸들
                           int amount                 // 복제 값 수
                           )
  {
//--- 오류 코드 재설정
   ResetLastError();
//--- 인덱스 0을 지닌 지표 버퍼로부터의 값으로 iVolumesBuffer 배열 일부를 채웁니다
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
     {
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iVolumes 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- iVolumesColors 배열의 일부를 인덱스 1이 있는 지표 버퍼의 값으로 채웁니다
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      //--- 복사가 실패하면 오류 코드를 알려줍니다
      PrintFormat("iVolumes 지표로부터 데이터 복제 실패, 오류 코드 %d",GetLastError());
      //--- 결과가 0인 종료 - 지표가 계산되지 않은 것으로 간주됨을 의미합니다
      return(false);
     }
//--- 모든 게 좋습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 지표 초기화 해제 함수                                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- 지표 삭제 후 차트 지우기
   Comment("");
  }
```

© 2000-2025, MetaQuotes Ltd.

최적화 결과 작업

최적화의 커스텀 프로세싱을 구성하는 함수는 전략 테스터로 귀결됩니다. 테스트 에이전트, 로컬에서의 Expert Advisor 및 스크립트에서 최적화 중 호출할 수 있습니다.

전략 테스터에서 Expert Advisor를 실행할 때 심플 타입이나 심플 스트럭쳐를 기반으로 고유한 데이터 배열을 생성할 수 있습니다(동적 배열의 문자열, 클래스 개체 또는 개체는 포함되지 않음). 이 데이터 세트는 프레임이라는 특수 구조에서 FrameAdd() 함수를 사용하여 저장할 수 있습니다. Expert Advisor를 최적화하는 동안 각 에이전트는 터미널에 일련의 프레임을 보낼 수 있습니다. 수신된 모든 프레임은 terminal_directory₩MQL5₩Files₩Tester 폴더의 Expert Advisor로 명명된 *.MQD 파일에 기록됩니다. 에이전트로부터 받은 순서대로 작성됩니다. 테스팅 에이전트에서 클라이언트 터미널의 프레임을 수신하면 TesterPass 이벤트가 생성됩니다.

프레임은 컴퓨터 메모리 및 지정된 이름의 파일에 저장할 수 있습니다. MQL5 언어는 프레임 수에 대한 제한을 설정하지 않습니다.

MQL5 Cloud Network의 메모리 및 디스크 공간 제한

다음 제한 사항은 MQL5 Cloud Network 최적화 작업 실행시에 적용됩니다.: Expert Advisor는 4GB 이상의 정보를 디스크에 쓰거나 4GB 이상의 RAM을 사용해서는 안 됩니다. 한도를 초과하면 네트워크 에이전트가 계산을 올바르게 완료할 수 없으며 열러분은 결과를 받을 수 없습니다. 그러나 계산에 소요된 모든 시간에 대해 비용이 청구됩니다.

각각의 최적화 단계에서 정보를 가져와야 하는 경우 디스크에 쓰지 말고 프레임을 보내세요 . MQL5 Cloud Network에서 계산하는 동안 Expert Advisors에서 파일 작업을 피하려면 다음과 같은 확인을 사용할 수 있습니다.

```
int handle=INVALID_HANDLE;
bool file_operations_allowed=true;
if(MQLInfoInteger(MQL_OPTIMIZATION) || MQLInfoInteger(MQL_FORWARD))
   file_operations_allowed=false;

if(file_operations_allowed)
  {
   ...
   handle=FileOpen(...);
   ...
  }
```

함수	기능
FrameFirst	프레임 리딩의 포인터를 처음으로 이동하고 이전에 설정된 필터를 재설정합니다.
FrameFilter	프레임 판독 필터를 설정하고 포인터를 시작 부분으로 이동합니다
FrameNext	프레임을 읽고 포인터를 다음 프레임으로 이동합니다
FrameInputs	프레임이 형성되는 입력 매개 변수를 수신합니다
FrameAdd	프레임을 데이터와 함께 추가합니다

© 2000-2025, MetaQuotes Ltd.

함수	기능
ParameterGetRange	Strategy Tester에서 Expert Advisor를 최적화할 때 입력 변수의 값 범위 및 변경 단계에 대한 데이터를 수신합니다.
ParameterSetRange	Strategy Tester에서 Expert Advisor를 최적화할 때 입력 변수(값, 변경 단계, 초기 및 최종 값)을 지정합니다.

참고 항목

테스트 통계, 실행 중인 MQL5 프로그램의 속성

© 2000-2025, MetaQuotes Ltd.

FrameFirst

프레임 리딩의 포인터를 처음으로 이동하고 설정된 필터를 재설정합니다.

```
bool  FrameFirst();
```

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출하십시오.

© 2000-2025, MetaQuotes Ltd.

FrameFilter

프레임 리딩 필터를 설정하고 포인터를 처음으로 이동합니다.

```
bool  FrameFilter(
   const string  name,          // Public name/label
   long          id             // 퍼블릭 ID
   );
```

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출하십시오.

참고

빈 문자열이 첫 번째 매개 변수로 전달되면 필터는 숫자 매개 변수에서만 작동합니다. 즉, 지정된 ID를 가진 프레임만 표시됩니다. 두번째 매개 변수의 값이 ULONG_MAX이면, 텍스트 필터만 작동합니다.

FrameFilter("", ULONG_MAX)의 호출은 FrameFirst(), 즉 필터를 사용하지 않는 것과 동일합니다.

© 2000-2025, MetaQuotes Ltd.

FrameNext

프레임을 읽고 포인터를 다음 프레임으로 이동합니다. 함수에는 두 가지 변수가 있습니다.

1. 하나의 숫자 값을 수신하기 위해 호출하는 중

```
bool  FrameNext(
   ulong&   pass,      // 프레임이 추가된 최적화의 패스 수
   string&  name,      // Public name/label
   long&    id,        // 퍼블릭 ID
   double&  value      // 값
   );
```

2. 프레임의 모든 데이터를 수신하기 위해 호출 중

```
bool  FrameNext(
   ulong&   pass,      // 프레임이 추가된 최적화의 패스 수
   string&  name,      // Public name/label
   long&    id,        // 퍼블릭 ID
   double&  value,     // 값
   void&    data[]     // 모든 타입의 배열
   );
```

Parameter

pass

[out] 전략 테스터의 최적화 중 통과 횟수.

name

[out] 식별자의 이름.

id

[out] 식별자의 값.

value

[out] 단일 숫자 값.

data

[out] 모든 종류의 배열.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출하십시오.

참고

두번째 버전의 요청에서는, *data[]* 배열에서 수신된 데이터를 올바르게 처리해야 합니다.

© 2000-2025, MetaQuotes Ltd.

FrameInputs

지정된 패스 넘버를 가진 프레임이 형성되는 입력 매개 변수를 수신합니다.

```
bool  FrameInputs(
   ulong    pass,                // 최적화에서 패스의 수
   string&  parameters[],        // "parameterN=valueN" 형태의 문자열 배열
   uint&    parameters_count     // 매개 변수의 총 수
   );
```

Parameter

pass

[in] 전략 테스터의 최적화 중 패스 숫자.

parameters

[out] 이름 및 매개 변수 값에 대한 설명이 있는 문자열 배열

parameters_count

[out] 배열 *parameters[]*의 요소 수.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출하십시오.

참고

*parameters_count*를 *parameters[]* 배열에서 얻었으므로, 루프를 구성하여 모든 레코드를 검색할 수 있습니다. 이렇게 하면 지정된 패스 번호에 대한 Expert Advisor의 입력 매개 변수 값을 찾는 데 도움이 됩니다.

© 2000-2025, MetaQuotes Ltd.

FrameAdd

데이터가 포함된 프레임을 추가합니다. 함수에는 두 가지 변수가 있습니다.

1. 파일에서 데이터를 추가합니다

```
bool  FrameAdd(
   const string  name,         // Public name/label
   long          id,           // 퍼블릭 ID
   double        value,        // 값
   const string  filename      // 데이터 파일의 이름
   );
```

2. 모든 유형의 배열에서 데이터 추가

```
bool  FrameAdd(
   const string  name,         // Public name/label
   long          id,           // 퍼블릭 ID
   double        value,        // 값
   const void&   data[]        // 모든 유형의 배열
   );
```

Parameter

name

[in] 퍼블릭 프레임 레이블. FrameFilter() 함수에서 필터에 사용할 수 있습니다.

id

[in] 프레임의 퍼블릭 식별자. FrameFilter() 함수에서 필터에 사용할 수 있습니다.

value

[in] 프레임에 쓸 숫자 값. OnTester() 함수에서와 같이 단일 패스 결과를 전송하는 데 사용됩니다.

filename

[in] 프레임에 추가할 데이터가 들어 있는 파일 이름. 파일이 다음과 같은 폴더에 있어야 합니다: MQL5/Files.

data

[in] 프레임에 쓸 모든 유형의 배열. 참조에 의해 통과됨

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면 GetLastError() 함수를 호출하십시오.

© 2000-2025, MetaQuotes Ltd.

ParameterGetRange

Strategy Tester에서 Expert Advisor를 최적화할 때 입력 변수의 값 범위 및 변경 단계에 대한 데이터를 수신합니다. 이 함수에는 두 가지 변수가 있습니다.

1. 정수 유형 입력 매개 변수에 대한 데이터 수신

```
bool  ParameterGetRange(
   const string  name,          // 매개 변수 (입력 변수) 명
   bool&         enable,        // 매개 변수 최적화 사용
   long&         value,         // 매개 변수 값
   long&         start,         // 초기 값
   long&         step,          // 단계 변경
   long&         stop           // 최종 값
   );
```

2. 실제 유형 입력 매개 변수에 대한 데이터 수신

```
bool  ParameterGetRange(
   const string  name,          // 매개 변수 (입력 변수) 명
   bool&         enable,        // 매개 변수 최적화 사용
   double&       value,         // 매개 변수 값
   double&       start,         // 초기 값
   double&       step,          // 단계 변경
   double&       stop           // 최종 값
   );
```

Parameter

name

[in] 입력 변수 ID. 이러한 변수는 응용프로그램의 외부 매개 변수입니다. 이 값은 차트에서 시작하거나 단일 테스트 중에 지정할 수 있습니다.

enable

[out] Strategy Tester에서 최적화 중 값을 열거하는 데 이 매개 변수를 사용할 수 있다는 플래그.

value

[out] 매개 변수 값.

start

[out] 최적화 중 초기 매개 변수 값.

step

[out] 값을 열거할 때 매개 변수 변경 단계.

stop

[out] 최적화 중 최종 매개 변수 값.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

참고

이 함수는 OnTesterInit(), OnTesterPass() 및 OnTesterDeinit() 핸들러에서만 호출할 수 있습니다. 이것은 Strategy Tester에서 최적화하는 동안 Expert Advisor의 입력 매개 변수 값과 변동 범위를 수신하기 위해 도입되었습니다.

OnTesterInit()에서 호출될 때, 가져온 데이터를 사용해 모든 입력 변수의 열거 규칙을 ParameterSetRange() 함수를 사용하여 재정의할 수 있습니다. 따라서 Strategy Tester 설정에 관계없이 새로운 Start, Stop 및 Step 값을 설정하고 입력 매개 변수를 최적화에서 완전히 제외할 수도 있습니다. 이를 통해 Expert Advisor의 주요 매개 변수 값에 따라 일부 매개 변수를 최적화에서 제외함으로써 최적화 중에 입력 매개 변수의 영역을 관리할 수 있습니다.

예:

```
#property description "ParameterGetRange() 함수 데모를 위한 Expert Advisor."
#property description "Strategy Tester의 최적화 모드에서 실행되어야 함"
//--- 입력 매개 변수
input int                  Input1=1;
input double               Input2=2.0;
input bool                 Input3=false;
input ENUM_DAY_OF_WEEK     Input4=SUNDAY;

//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Expert Advisor는 Strategy Tester에서만 작동하도록 설계되었습니다
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("Strategy Tester의 최적화 모드에서 실행되어야 함!");
      //--- Expert Advisor 작업을 미리 완료하고 차트에서 제거합니다.
      return(INIT_FAILED);
     }
//--- 초기화의 성공적 완료
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- long 유형 입력 매개 변수 예제
   string name="Input1";
   bool enable;
   long par1,par1_start,par1_step,par1_stop;
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
   Print("첫번째 매개변수");
   PrintFormat("%s=%d  enable=%s  from %d to %d with step=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- double 유형 입력 매개 변수 예제
```

© 2000-2025, MetaQuotes Ltd.

```
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("두번째 매개변수");
   PrintFormat("%s=%G  enable=%s  from %G to %G with step=%G",
               name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- example for bool type input parameter
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("세번째 매개변수");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- 열거형 입력 매개 변수 예제
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("네번째 매개변수");
   PrintFormat("%s=%s  enable=%s  from %s to %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
  }
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- 최적화 완료 후 이 메시지가 표시됩니다.
   Print(__FUNCTION__," Optimization completed");
  }
```

© 2000-2025, MetaQuotes Ltd.

ParameterSetRange

Strategy Tester에서 Expert Advisor를 최적화할 때 입력 변수(값, 변경 단계, 초기 및 최종 값)를 지정합니다. 이 함수에는 두 가지 변수가 있습니다.

1. 정수 유형 입력 매개 변수의 값 지정

```
bool  ParameterSetRange(
   const string  name,          // 매개 변수 (입력 변수) 명
   bool          enable,        // 매개 변수 최적화 사용
   long          value,         // 매개 변수 값
   long          start,         // 초기값
   long          step,          // 단계 변경
   long          stop           // 최종값
   );
```

2. real 유형 입력 매개 변수의 값 지정

```
bool  ParameterSetRange(
   const string  name,          // 매개 변수 (입력 변수) 명
   bool          enable,        // 매개 변수 최적화 사용
   double        value,         // 매개 변수 값
   double        start,         // 초기값
   double        step,          // 단계 변경
   double        stop           // 최종값
   );
```

Parameter

name

[in] input or sinput variable ID. 이러한 변수는 응용프로그램의 외부 매개 변수입니다. 프로그램을 시작할 때 값을 지정할 수 있습니다.

enable

[in] Strategy Tester에서 최적화 중 값을 열거하려면 이 매개 변수 활성화하십시오.

value

[in] 매개 변수 값.

start

[in] 최적화 중 초기 매개 변수 값.

step

[in] 값을 열거할 때 매개 변수 변경 단계.

stop

[in] 최적화 중 최종 매개 변수 값.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

참고

이 기능은 Strategy Tester에서 최적화를 시작할 때 OnTesterInit() 핸들러에서만 호출할 수 있습니다. 매개 변수의 범위 및 변경 단계를 지정하도록 설계되었습니다. 매개 변수는 Strategy Tester 설정에 관계없이 최적화에서 완전히 제외할 수 있습니다. 또한 최적화 프로세스에서 입력 수정자와 함께 선언된 변수를 사용할 수 있습니다.

ParameterSetRange() 함수를 사용하면 필수 입력 매개 변수를 최적화에서 포함하거나 제외하고 필요한 범위 및 변경 단계를 설정하여 핵심 매개 변수의 값에 따라 Strategy Tester에서 Expert Advisor 최적화를 관리할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

이벤트 작업

이 그룹에는 사용자 지정 이벤트 및 타이머 이벤트 작업을 위한 기능이 포함되어 있습니다. 이 그룹 외에도 사전 정의된 이벤트를 처리하는 특수 기능이 있습니다.

기능	액션
EventSetMillisecondTimer	현재 차트에 대해 1초 미만의 주기를 가진 고해상도 타이머의 이벤트 생성기를 시작
EventSetTimer	현재 차트에 대해 지정된 주기로 타이머 이벤트 생성기를 시작
EventKillTimer	현재 차트에서 타이머에 의한 이벤트 생성을 중지
EventChartCustom	지정된 차트에 대한 사용자 지정 이벤트를 성성

추가 참조

차트 이벤트 유형

© 2000-2025, MetaQuotes Ltd.

EventSetMillisecondTimer

이 기능은 이 엑스퍼트 어드바이저 또는 지표에 대해 타이머 이벤트를 1초 미만의 간격으로 생성해야 함을 클라이언트 터미널에 나타냅니다.

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // 밀리초의 수
   );
```

매개변수

milliseconds

[in] 타이머 이벤트의 빈도를 정의하는 밀리초.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 받으려면 GetLastError() 함수를 호출해야 합니다.

주의

이 기능은 타이머가 필요한 경우에 맞게 고안되었습니다. 즉, 타이머 이벤트는 초당 1회보다 더 자주 수신해야 합니다. 주기가 1초 이상인 일반 타이머로 충분하면 EventSetTimer()를 사용하십시오.

일반적으로 타이머 시간이 줄어들면 타이머 이벤트 처리기가 더 자주 호출되므로 테스트 시간이 늘어납니다. 실시간 모드에서 작업하는 경우 하드웨어 제한으로 인해 타이머 이벤트가 10~16 밀리초 내에 1회 이상 생성되지 않습니다.

일반적으로 이 기능은 OnInit() 함수 또는 클래스 생성자에서 호출해야 합니다. 타이머에서 발생하는 이벤트를 처리하려면 엑스퍼트 어드바이저 또는 지표에 OnTimer() 기능이 있어야 합니다.

각 엑스퍼트 어드바이저와 지표는 이 타이머에서만 이벤트를 수신하는 자체 타이머로 작동합니다. mql5 애플리케이션 종료 중에 타이머가 생성되었지만 EventKillTimer() 기능으로 비활성화 되지 않은 경우 타이머가 강제로 폐기됩니다.

각 프로그램에 대해 타이머를 하나만 시작할 수 있습니다. 각 mql5 애플리케이션 및 차트에는 새로 도착한 모든 이벤트가 배치되는 자체 이벤트 대기열이 있습니다. 대기열에 이미 타이머 이벤트가 포함되어 있거나 이벤트 처리 단계에 있는 경우 새 타이머 이벤트가 mql5 애플리케이션 대기열에 추가되지 않습니다.

© 2000-2025, MetaQuotes Ltd.

EventSetTimer

이 기능은 클라이언트 터미널에 이 지표 또는 엑스퍼트 어드바이저의 경우 지정된 주기로 타이머의 이벤트를 생성해야 함을 나타냅니다.

```
bool  EventSetTimer(
   int  seconds      // 초의 수
   );
```

매개변수

seconds

[in] 타이머 이벤트 발생 빈도를 결정하는 시간(초).

값 반환

성공하는 경우 true, 그렇지 않은 경우 false. 오류 코드를 가져오려면 GetLastError() 함수를 호출해야 합니다.

주의

일반적으로 이 기능은 OnInit() 함수 또는 클래스생성자에서 호출해야 합니다. 타이머에서 발생하는 이벤트를 처리하기 위해서는 엑스퍼트 어드바이저가 OnTimer() 기능을 가지고 있어야 합니다.

모든 엑스퍼트 어드바이저와 모든 지표는 고유한 타이머로 작동하며, 이 타이머에서만 이벤트를 수신합니다. mql5 프로그램의 작동이 중지되는 즉시 타이머가 생성되었지만 EventKillTimer() 기능에 의해 비활성화가 되지 않은 경우 타이머가 강제로 폐기됩니다.

각 프로그램에 대해 타이머를 두 개 이상 실행할 수 없습니다. 각 mql5 프로그램과 각 차트에는 새로 수신된 모든 이벤트가 배치되는 자체 이벤트 대기열이 있습니다. 타이머 이벤트가 대기열에 있거나 처리 중인 경우 새 타이머 이벤트는 mql5 프로그램의 대기열에 배치되지 않습니다.

© 2000-2025, MetaQuotes Ltd.

EventKillTimer

이벤트 생성을 중지하는데 필요한 클라이언트 터미널을 타이머에서 지정.

```
void  EventKillTimer();
```

값 반환

반환 값 없음.

주의

일반적으로 이 함수는 OnDeinit()에서 EventSetTimer() 함수가 호출된 경우OnInit()함수에서 호출되어야 합니다. EventSetTimer() 기능이 이 클래스의 생성자에서 호출된 경우 클래스 소멸기에서 이 함수를 호출할 수도 있습니다.

모든 엑스퍼트 어드바이저와 모든 지표는 고유한 타이머로 작동하며, 이 타이머에서만 이벤트를 수신합니다. mql5 프로그램의 작동이 중지되는 즉시 타이머가 생성되었지만 EventKillTimer() 기능에 의해 비활성화 되지 않은 경우 타이머가 강제로 폐기됩니다

© 2000-2025, MetaQuotes Ltd.

EventChartCustom

함수는 지정된 차트에 대한 사용자 지정 이벤트를 생성.

```
bool  EventChartCustom(
   long    chart_id,            // 이벤트 수신 차트의 식별자
   ushort  custom_event_id,     // 이벤트 식별자
   long    lparam,              // long 유형의 매개변수
   double  dparam,              // double 유형의 매개변수
   string  sparam               // 이벤트의 string 매개변수
   );
```

매개변수

chart_id

[in] 차트 식별자. 0은 현재 차트를 의미합니다.

custom_event_id

[in] 사용자 이벤트의 ID. 이 식별자는 CHARTEVENT_CUSTOM에 자동으로 추가되고 정수 유형으로 변환됩니다.

lparam

[in] OnChartEvent 기능으로 전달된 long 유형의 이벤트 매개변수.

dparam

[in] OnChartEvent 기능으로 전달된 double 유형의 이벤트 매개변수.

sparam

[in] OnChartEvent 기능으로 전달된 string 유형의 이벤트 매개변수. 문자열이 63자를 초과하면 잘립니다.

값 반환

이벤트를 수신하는 차트의 이벤트 대기열에 사용자 지정 이벤트가 성공적으로 배치된 경우 true를 반환합니다. 실패한 경우 false를 반환합니다. 오류 코드를 가져오려면 GetLastError()를 사용.

주의

지정된 차트에 연결된 엑스퍼트 어드바이저 또는 지표는 OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam) 기능을 사용하여 이벤트를 처리합니다.

각 이벤트 유형에 대해 OnChartEvent() 기능의 입력 매개변수는 이 이벤트를 처리하는 데 필요한 한정된 값을 가집니다. 이 매개변수를 통해 전달된 이벤트와 값이 아래 표에 나열되어 있습니다.

이벤트	id 매개변수의 값	lparam 매개변수의 값	dparam 매개변수의 값	sparam 매개변수의 값
키입력 이벤트	CHARTEVENT_KEYDOWN	눌려진 키의 코드	반복 횟수 (사용자가 키를 누를 상태로 인해 키 입력이 반복되는 횟수)	키보드 버튼의 상태를 설명하는 비트 마스크의 문자열 값

© 2000-2025, MetaQuotes Ltd.

이벤트	id 매개변수의 값	lparam 매개변수의 값	dparam 매개변수의 값	sparam 매개변수의 값
마우스 이벤트 (속성 CHART_EVENT_MOUSE_MOVE=true 이 차트에 설정된 경우)	CHARTEVENT_MOUSE_MOVE	X 좌표	Y 좌표	마우스 버튼의 상태를 설명하는 비트 마스크의 문자열 값
그래픽 개체 생성 이벤트 (CHART_EVENT_OBJECT_CREATE =true 가 차트에 설정된 경우)	CHARTEVENT_OBJECT_CREATE	—	—	생성된 그래픽 개체 이름
속성 대화 상자를 통한 개체 속성 변경 이벤트	CHARTEVENT_OBJECT_CHANGE	—	—	수정된 그래픽 개체의 이름
그래픽 개체 삭제 이벤트 (CHART_EVENT_OBJECT_DELETE =true 가 차트에 설정된 경우)	CHARTEVENT_OBJECT_DELETE	—	—	삭제된 그래픽 개체의 이름
차트를 마우스로 클릭하는 경우	CHARTEVENT_CLICK	X 좌표	Y 좌표	—
차트에 속한 그래픽 개체에서 마우스를 클릭하는 이벤트	CHARTEVENT_OBJECT_CLICK	X 좌표	Y 좌표	이벤트가 발생한 그래픽 개체 이름
마우스를 사용하여 그래픽 개체를 끄는 이벤트	CHARTEVENT_OBJECT_DRAG	—	—	이동된 그래픽 개체의 이름
LabelEdit 그래픽 개체의 입력 상자에서 완료된 텍스트 편집 이벤트	CHARTEVENT_OBJECT_ENDEDIT	—	—	텍스트 편집이 완료된 LabelEdit 그래픽 개체의 이름
차트 변경 이벤트	CHARTEVENT_CHART_CHANGE	—	—	—
N 번 아래의 사용자 이벤트 ID	CHARTEVENT_CUSTOM+N	EventChartCustom() 함수에 의해 설정된 값	EventChartCustom() 함수에 의해 설정된 값	EventChartCustom() 함수에 의해 설정된 값

예를 들어:

```
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                ButtonClickExpert.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

string buttonID="Button";
string labelID="Info";
int broadcastEventID=5000;
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 함수                                             |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 사용자 지정 이벤트를 전송하기 위한 버튼 생성
   ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
   ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
   ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
   ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
   ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
   ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
   ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
   ObjectSetString(0,buttonID,OBJPROP_TEXT,"Button");
   ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
   ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- 정보 표시를 위한 레이블 생성
   ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
   ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
   ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
   ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
   ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
   ObjectSetString(0,labelID,OBJPROP_TEXT,"No information");
   ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
   ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 초기화 해제 함수                                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
```

© 2000-2025, MetaQuotes Ltd.

```
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------+
//| 엑스퍼트 틱 함수                                                 |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- 마우스 버튼을 눌러 이벤트를 확인
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      string clickedChartObject=sparam;
      //--- 이름이 buttonID인 개체를 클릭하는 경우
      if(clickedChartObject==buttonID)
        {
         //--- 버튼 상태 - 누르거나 누르지 않음
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- 디버그 메시지를 기록
         Print("버튼 눌림 = ",selected);
         int customEventID; // 전송할 사용자 지정 이벤트 수
         string message;    // 이벤트에 전송될 메시지
         //--- 버튼이 눌린 경우
         if(selected)
           {
            message="Button pressed";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // 버튼이 눌리지 않음
           {
            message="Button in not pressed";
            customEventID=CHARTEVENT_CUSTOM+999;
           }
         //--- 사용자 지정 이벤트 "당사" 차트 전송
         EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
         ///--- 열려 있는 모든 차트에 메시지 전송
         BroadcastEvent(ChartID(),0,"Broadcast Message");
         //--- 디버그 메시지
         Print("ID를 사용하여 이벤트 전송 = ",customEventID);
        }
      ChartRedraw();// 모든 차트 개체를 강제로 다시 그리기
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 이벤트가 사용자 이벤트세 속하는지 확인
   if(id>CHARTEVENT_CUSTOM)
     {
      if(id==broadcastEventID)
        {
         Print("Got broadcast message from a chart with id = "+lparam);
        }
      else
        {
         //--- 이벤트에서 문자 메시지를 읽었습니다
         string info=sparam;
         Print("ㅣD로 사용자 이벤트 처리 = ",id);
         //--- 레이블에 메시지 표시
         ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
         ChartRedraw();// 모든 차트 개체를 강제로 다시 그리기
        }
     }
  }
//+------------------------------------------------------------------+
//| 열린 모든 차트에 브로드캐스트 이벤트 전송                          |
//+------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
  {
   int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
   long currChart=ChartFirst();
   int i=0;
   while(i<CHARTS_MAX)                 // CHARTS_MAX 오픈 차트 밖에 없습니다
     {
      EventChartCustom(currChart,eventID,lparam,dparam,sparam);
      currChart=ChartNext(currChart); // 이전 차트로부터 새 차트를 받았습니다
      if(currChart==-1) break;        // 차트 목록의 끝에 도달함
      i++;// 카운터를 늘리는 것을 잊지 마십시오
     }
  }
//+------------------------------------------------------------------+
```

추가 참조

클라이언트 터미널의 이벤트, 이벤트 처리기 기능

© 2000-2025, MetaQuotes Ltd.

OpenCL 작업

OpenCL프로그램은 OpenCL 1.1이상을 지원하는 비디오 카드에 계산을 수행하기 위해 사용됩니다. 최신 비디오 카드에는 수신 데이터 스트림과 함께 간단한 수학적 연산을 동시에 수행할 수 있는 수백 개의 소형 전문 프로세서가 포함되어 있습니다. OpenCL 언어는 병렬 컴퓨팅을 구성하고 특정 종류의 작업에 더 빠른 속도를 제공합니다.

일부 그래픽 카드에서는 기본적으로 double 유형 번호로 작업할 수 있습니다. 이로 인해 컴파일 오류 5105가 발생할 수 있습니다. double 유형 번호에 대한 지원을 활성화하려면 OpenCL 프로그램에 #pragma OPENCL EXTENSION cl_khr_fp64 : enable 명령을 추가하십시오. 그러나 그래픽 카드가 double을 지원하지 않는 경우 이 명령문을 활성화하는 것은 도움이 되지 않습니다.

OpenCL에 대한 소스 코드를 별도의 CL 파일에 쓰는 것이 좋습니다. 이 소스 코드는 나중에 리소스 변수를 사용하여 MQL5 프로그램에 포함될 수 있습니다.

OpenCL프로그램에서 핸들링 에러

OpenCL 프로그램에서 발생한 마지막 오류에 대한 정보를 얻으려면CLGetInfoInteger및CLGetInfoString함수를 사용하십시오. 이 함수들은 오류 코드와 설명을 가져옵니다.

OpenCL 마지막 에러 코드: 최근에 발생한 OpenCL 에러를 알아보기 위해서는 CLGetInfoInteger를 호출하십시오, 호출은 *handle* 매개변수가 무시되는 동안 이루어져야 합니다 (0으로 설정 가능). 에러 설명 :https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS.

알 수 없는 오류 코드의 경우 "알 수 없는 OpenCL 오류 N"이란 문자열이 반환되며 여기서 N은 오류 코드입니다. 예:

```
//--- 첫 번째 'handle' 매개변수는 마지막 오류를 가져올 때 무시됩니다.code
intcode= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
```

OpenCL 오류에 대한 텍스트 설명: 최근의 OpenCL 에러를 가져오려면 CLGetInfoString을 호출하세요. 에러 코드는 반드시 *핸들*매개변수로 전달되어야 합니다.

에러 설명:https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. 에러 코드 대신 CL_LAST_ERROR가 전달되면 함수는 마지막 에러에 대한 설명을 반환합니다. 예시:

```
//--- 마지막 OpenCL 에러 코드 가져오기
int    code= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
stringdesc;// 에러 설명

//--- 에러에 대한 설명을 얻으려면 에러 코드를 사용하세요
if(!CLGetInfoString(code,CL_ERROR_DESCRIPTION,desc))
 desc="OpenCL 에러 설명을 얻을 수 없음"+ (string)GetLastError();
Print(desc);

//--- 코드를 먼저 가져오지 않으면서 마지막 OpenCL 에러에 대한 설명을 가져오려면 CL_LAST_ERROR를
if(!CLGetInfoString(CL_LAST_ERROR,CL_ERROR_DESCRIPTION,desc))
 desc="OpenCL 에러 설명을 얻을 수 없음"+ (string)GetLastError();
Print(desc);;
```

© 2000-2025, MetaQuotes Ltd.

지금까지는 내부 열거형의 이름이 에러에 대한 설명으로 제공되었습니다. 여기에서 디코딩을 찾을 수 있습니다:https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. 예를 들어 CL_INVALID_KERNEL_ARGS 값은 "일부 커널 인수가 설정되지 않았거나 유효하지 않은 경우 커널을 큐에 더할 때 반환됨"을 의미합니다.

OpenCL에서 프로그램을 실행하는 함수:

함수	기능
CLHandleType	ENUM_OPENCL_HANDLE_TYPE 열거값으로 OpenCL 핸들 유형을 반환합니다
CLGetInfoInteger	OpenCL 개체 또는 디바이스에 대한 정수 속성 값을 반환합니다
CLContextCreate	OpenCL 컨텍스트를 생성합니다
CLContextFree	OpenCL 컨텍스트를 제거합니다
CLGetDeviceInfo	OpenCL 드라이버로부터 장치 속성 수신
CLProgramCreate	소스 코드에서 OpenCL 프로그램을 생성합니다
CLProgramFree	OpenCL 프로그램을 제거합니다
CLKernelCreate	OpenCL 시작 함수를 생성합니다
CLKernelFree	OpenCL 시작 기능을 제거합니다
CLSetKernelArg	OpenCL 함수에 대한 매개 변수를 설정합니다
CLSetKernelArgMem	OpenCL 버퍼를 OpenCL 함수의 매개 변수로 설정합니다
CLSetKernelArgMemLocal	로컬 버퍼를 커널 함수의 인수로 설정합니다
CLBufferCreate	OpenCL 버퍼를 생성합니다
CLBufferFree	OpenCL 버퍼를 삭제합니다
CLBufferWrite	OpenCL 버퍼에 배열을 씁니다
CLBufferRead	OpenCL 버퍼를 배열로 읽습니다
CLExecute	OpenCL 프로그램을 실행합니다
CLExecutionStatus	OpenCL 프로그램 실행 상태를 반환합니다

참고 항목

OpenCL, Resources

© 2000-2025, MetaQuotes Ltd.

CLHandleType

ENUM_OPENCL_HANDLE_TYPE 열거값으로 OpenCL 핸들 유형을 반환합니다.

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // OpenCL 개체의 핸들
   );
```

Parameter

핸들

[in] OpenCL 객체에 대한 핸들: 컨텍스트, 커널 또는 OpenCL 프로그램.

반환값

ENUM_OPENCL_HANDLE_TYPE 열거값으로 사용되는 OpenCL 핸들 유형.

ENUM_OPENCL_HANDLE_TYPE

식별자	설명
OPENCL_INVALID	잘못된 핸들
OPENCL_CONTEXT	OpenCL 컨텍스트 핸들
OPENCL_PROGRAM	OpenCL 프로그램 핸들
OPENCL_KERNEL	OpenCL kernel 핸들
OPENCL_BUFFER	OpenCL 버퍼 핸들

© 2000-2025, MetaQuotes Ltd.

CLGetInfoInteger

OpenCL 개체 또는 디바이스에 대한 정수 속성 값을 반환합니다.

```
long  CLGetInfoInteger(
   int  handle,                          // OpenCL 개체의 핸들 또는 OpenCL 장치의 번호
   ENUM_OPENCL_PROPERTY_INTEGER  prop    // 요청된 속성
   );
```

Parameter

handle

[in] OpenCL 개체의 핸들 또는 OpenCL 장치의 번호. OpenCL 장치의 번호 매기는 0부터 시작합니다.

prop

[in] ENUM_OPENCL_PROPERTY_INTEGER 열거값에서, 요청한 속성의 유형이며, 가져올 값입니다.

반환값

성공한 경우 속성의 값 또는 오류가 발생한 경우 -1입니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

ENUM_OPENCL_PROPERTY_INTEGER

식별자	설명	유형
CL_DEVICE_COUNT	OpenCL이 지원되는 장치 수. 이 속성은 첫 번째 매개 변수를 지정할 필요가 없는데, 즉, *핸들* 매개 변수에 0 값을 전달할 수 있습니다.	int
CL_DEVICE_TYPE	장치 유형	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	고유 벤더 식별자	uint
CL_DEVICE_MAX_COMPUTE_UNITS	OpenCL 장치에서 병렬로 연산된 작업 수입니다. 하나의 작업 그룹이 하나의 연산 작업을 해결합니다. 최소 값은 1입니다	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	장치의 최고 설정 주파수(MHz) 입니다.	uint
CL_DEVICE_GLOBAL_MEM_SIZE	장치의 글로벌 메모리 크기(바이트)	ulong
CL_DEVICE_LOCAL_MEM_SIZE	처리된 데이터 (scene) 로컬 메모리의 크기(바이트)	uint
CL_BUFFER_SIZE	OpenCL 버퍼의 실제 크기(바이트)	ulong
CL_DEVICE_MAX_WORK_GROUP_SIZE	OpenCL 장치에 사용할 수 있는 로컬 작업 그룹의 총 수입니다.	ulong

© 2000-2025, MetaQuotes Ltd.

식별자	설명	유형
CL_KERNEL_WORK_GROUP_SIZE	OpenCL 프로그램에 사용할 수 있는 총 로컬 작업 그룹 수.	ulong
CL_KERNEL_LOCAL_MEM_SIZE	그룹의 모든 병렬 작업을 해결하기 위해 OpenCL 프로그램에서 사용하는 로컬 메모리 크기(바이트). CL_DEVICE_LOCAL_MEM_SIZE를 사용하여 가능한 최대 값을 수신합니다	ulong
CL_KERNEL_PRIVATE_MEM_SIZE	OpenCL 프로그램 커널의 각 작업에서 사용하는 개인 메모리의 최소 크기(바이트).	ulong
CL_LAST_ERROR	마지막 OpenCL 에러의 값	int

The ENUM_CL_DEVICE_TYPE 열거값에 OpenCL을 지원하는 가능한 장치 유형이 포함되어 있습니다. 장치 번호를 기준으로 장치 유형을 받거나 OpenCL 개체의 핸들을 CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE)를 호출하여 받을 수 있습니다.

ENUM_CL_DEVICE_TYPE

식별자	설명
CL_DEVICE_ACCELERATOR	전용 OpenCL 가속기 (예: IBM CELL Blade). 이러한 디바이스는 PCIe와 같은 주변 상호 연결을 사용하여 호스트 프로세서와 통신합니다.
CL_DEVICE_CPU	호스트 프로세서인 OpenCL 장치. 호스트 프로세서는 OpenCL 구현을 실행하고 단일 또는 다중 코어 CPU입니다.
CL_DEVICE_GPU	GPU인 OpenCL 장치.
CL_DEVICE_DEFAULT	시스템에서의 기본 OpenCL 장치. 기본 장치는 CL_DEVICE_TYPE_CUSTOMM 장치일 수 없습니다.
CL_DEVICE_CUSTOM	OpenCL C로 작성된 프로그램을 지원하지 않는 전용 가속기.

예:

```
void OnStart()
  {
   int cl_ctx;
//--- OpenCL 컨텍스트 초기화
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL을 찾을 수 없습니다");
      return;
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- OpenCL 장치에 대한 일반 정보 표시
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY)," MHz"
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE)," by
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE)," byte
//--- free OpenCL 컨텍스트
   CLContextFree(cl_ctx);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLGetInfoString

OpenCL 개체 또는 디바이스에 대한 속성의 문자열 값을 반환합니다.

```
bool  CLGetInfoString(
   int  handle,                              // OpenCL 개체 핸들 또는 OpenCL 장치 번호
   ENUM_OPENCL_PROPERTY_STRING  prop,        // 요청된 속성
   string&  value                            // 참조 문자열
   );
```

Parameter

handle

[in] OpenCL 개체 핸들 또는 OpenCL 장치 번호. OpenCL 장치의 넘버링은 0부터 시작합니다.

prop

[in] ENUM_OPENCL_PROPERTY_STRING 열거값에서 요청된 속성의 유형이며, 값을 구해야 합니다.

value

[out] 속성 값을 수신하는 문자열.

반환값

성공적이면 true, 아니면 false. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

ENUM_OPENCL_PROPERTY_STRING

식별자	설명
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE - OpenCL 프로파일. 프로파일 이름은 다음 값 중 하나일 수 있습니다. • FULL_PROFILE - 구현은 OpenCL을 지원합니다(기능은 OpenCL 지원을 위한 추가 확장 없이 커널 규격의 일부로 정의됨). • EMBEDDED_PROFILE - 구현은 추가적으로 OpenCL을 지원합니다. 수정된 프로파일은 각 OpenCL 버전에 대한 하위 집합으로 정의됩니다.
CL_PLATFORM_VERSION	OpenCL 버전
CL_PLATFORM_VENDOR	플랫폼 벤더 이름
CL_PLATFORM_EXTENSIONS	플랫폼에서 지원되는 확장 목록. 이 플랫폼과 관련된 모든 장치에서 확장명을 지원해야 합니다.
CL_DEVICE_NAME	장치명
CL_DEVICE_VENDOR	벤더명
CL_DRIVER_VERSION	major_number.minor_number 형식의 OpenCL 드라이버 버전
CL_DEVICE_PROFILE	OpenCL 디바이스 프로파일. 프로파일 이름은 다음 값 중 하나일 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

식별자	설명
	• FULL_PROFILE – 구현은 OpenCL을 지원합니다(기능은 OpenCL 지원을 위한 추가 확장 없이 커널 규격의 일부로 정의됨). • EMBEDDED_PROFILE – 구현은 추가적으로 OpenCL을 지원합니다. 수정된 프로파일은 각 OpenCL 버전에 대한 하위 집합으로 정의됩니다.
CL_DEVICE_VERSION	"OpenCL<space><major_version.minor_version><space><vendor-specific information>" 포맷의 OpenCL 버전
CL_DEVICE_EXTENSIONS	장치에서 지원하는 확장 목록. 이 목록에는 벤더가 지원하는 확장명과 하나 이상의 승인된 이름이 포함될 수 있습니다. cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_gl_sharing cl_khr_gl_event cl_khr_d3d10_sharing cl_khr_dx9_media_sharing cl_khr_d3d11_sharing
CL_DEVICE_BUILT_IN_KERNELS	지원되는 커널 목록은 ";"로 구분됩니다.
CL_DEVICE_OPENCL_C_VERSION	이 장치에 대해 컴파일러가 지원하는 최대 버전. 버전 포맷: "OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> "
CL_ERROR_DESCRIPTION	OpenCL 에러에 대한 설명

예:

```
void OnStart()
  {
   int cl_ctx;
   string str;
//--- OpenCL 컨텍스트 초기화
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL을 찾을 수 없습니다");
      return;
     }
//--- 플랫폼에 대한 정보 표시
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("OpenCL platform name: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("OpenCL platform vendor: ",str);
```

© 2000-2025, MetaQuotes Ltd.

```
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("OpenCL platform ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("OpenCL platform profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("OpenCL platform ext: ",str);
//--- 장치에 대한 정보 표시
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("OpenCL device name: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("OpenCL device profile: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("OpenCL device kernels: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("OpenCL device ext: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("OpenCL device vendor: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("OpenCL device ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("OpenCL open c ver: ",str);
//--- OpenCL 장치에 대한 일반 정보 표시
   Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
   Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
   Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//--- free OpenCL 컨텍스트
   CLContextFree(cl_ctx);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLContextCreate

OpenCL 컨텍스트를 생성하고 핸들을 반환합니다.

```
int  CLContextCreate(
   int  device=CL_USE_ANY     // OpenCL 장치 또는 매크로의 일련 번호
   );
```

매개변수

device

[in] 시스템에 있는 OpenCL-장치의 순서 번호. 특정 숫자 대신 다음 값 중 하나를 지정할 수 있습니다.

- CL_USE_ANY – OpenCL이 지원되는 모든 사용 가능한 장치가 허용됩니다;
- CL_USE_CPU_ONLY – CPU에서만 OpenCL 에뮬레이션이 허용됩니다;
- CL_USE_GPU_ONLY – OpenCL 에뮬레이션은 금지되며 OpenCL 지원(비디오 카드)이 있는 특수 장치만 사용할 수 있습니다;
- **CL_USE_GPU_DOUBLE_ONLY** - double 유형을 지원하는 GPU만 허용됩니다.

반환값

성공한 경우 OpenCL 컨텍스트에 대한 핸들, 그렇지 않은 경우 -1. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

CLContextFree

OpenCL 컨텍스트를 제거합니다.

```
void  CLContextFree(
   int  context     // OpenCL 컨텍스트 처리
   );
```

Parameter

context

[in] OpenCL 컨텍스트 처리.

반환값

없음. 내부 오류의 경우 _LastError의 값이 변경됩니다. 오류에 대해서는 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

CLGetDeviceInfo

이 함수는 OpenCL 드라이버로부터 장치 속성을 수신합니다.

```
bool  CLGetDeviceInfo(
   int     handle,          // OpenCL 장치 핸들
   int     property_id,     // 요청된 속성 ID
   uchar&  data[],          // 데이터 수신을 위한 배열
   uint&   size             // 배열 요소의 이동, 기본값은 0.
   );
```

Parameter

handle

[in] CLContextCreate() 함수로 만든 OpenCL 장치 인덱스 또는 OpenCL 핸들.

property_id

[in] 수신해야 하는 OpenCL 장치 속성의 ID. 값은 아래 표에 나열된 미리 결정된 값 중 하나일 수 있습니다.

data[]

[out] 요청한 속성에서 데이터를 수신하기 위한 어레이.

size

[out] *data[]* 배열에서 수신된 데이터의 크기.

반환값

성공적이면 true, 아니면 false. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

참고

1차원 배열의 경우, OpenCL 버퍼에 대한 데이터 판독이 시작되는 요소의 수는 AS_SERIES 플래그를 고려하여 계산됩니다.

OpenCL 장치 속성의 사용 가능한 ID 목록

속성과 함수에 대한 정확한 설명은 OpenCL 공식 웹사이트에서 확인할 수 있습니다..

식별자	값
CL_DEVICE_TYPE	0x1000
CL_DEVICE_VENDOR_ID	0x1001
CL_DEVICE_MAX_COMPUTE_UNITS	0x1002
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS	0x1003
CL_DEVICE_MAX_WORK_GROUP_SIZE	0x1004
CL_DEVICE_MAX_WORK_ITEM_SIZES	0x1005
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006

© 2000-2025, MetaQuotes Ltd.

식별자	값
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT	0x1007
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT	0x1008
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG	0x1009
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT	0x100A
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE	0x100B
CL_DEVICE_MAX_CLOCK_FREQUENCY	0x100C
CL_DEVICE_ADDRESS_BITS	0x100D
CL_DEVICE_MAX_READ_IMAGE_ARGS	0x100E
CL_DEVICE_MAX_WRITE_IMAGE_ARGS	0x100F
CL_DEVICE_MAX_MEM_ALLOC_SIZE	0x1010
CL_DEVICE_IMAGE2D_MAX_WIDTH	0x1011
CL_DEVICE_IMAGE2D_MAX_HEIGHT	0x1012
CL_DEVICE_IMAGE3D_MAX_WIDTH	0x1013
CL_DEVICE_IMAGE3D_MAX_HEIGHT	0x1014
CL_DEVICE_IMAGE3D_MAX_DEPTH	0x1015
CL_DEVICE_IMAGE_SUPPORT	0x1016
CL_DEVICE_MAX_PARAMETER_SIZE	0x1017
CL_DEVICE_MAX_SAMPLERS	0x1018
CL_DEVICE_MEM_BASE_ADDR_ALIGN	0x1019
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE	0x101A
CL_DEVICE_SINGLE_FP_CONFIG	0x101B
CL_DEVICE_GLOBAL_MEM_CACHE_TYPE	0x101C
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE	0x101D
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE	0x101E
CL_DEVICE_GLOBAL_MEM_SIZE	0x101F
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE	0x1020
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022

© 2000-2025, MetaQuotes Ltd.

식별자	값
CL_DEVICE_LOCAL_MEM_SIZE	0x1023
CL_DEVICE_ERROR_CORRECTION_SUPPORT	0x1024
CL_DEVICE_PROFILING_TIMER_RESOLUTION	0x1025
CL_DEVICE_ENDIAN_LITTLE	0x1026
CL_DEVICE_AVAILABLE	0x1027
CL_DEVICE_COMPILER_AVAILABLE	0x1028
CL_DEVICE_EXECUTION_CAPABILITIES	0x1029
CL_DEVICE_QUEUE_PROPERTIES	0x102A
CL_DEVICE_NAME	0x102B
CL_DEVICE_VENDOR	0x102C
CL_DRIVER_VERSION	0x102D
CL_DEVICE_PROFILE	0x102E
CL_DEVICE_VERSION	0x102F
CL_DEVICE_EXTENSIONS	0x1030
CL_DEVICE_PLATFORM	0x1031
CL_DEVICE_DOUBLE_FP_CONFIG	0x1032
CL_DEVICE_PREFERRED_VECTOR_WIDTH_HALF	0x1034
CL_DEVICE_HOST_UNIFIED_MEMORY	0x1035
CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR	0x1036
CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT	0x1037
CL_DEVICE_NATIVE_VECTOR_WIDTH_INT	0x1038
CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG	0x1039
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT	0x103A
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE	0x103B
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF	0x103C
CL_DEVICE_OPENCL_C_VERSION	0x103D
CL_DEVICE_LINKER_AVAILABLE	0x103E
CL_DEVICE_BUILT_IN_KERNELS	0x103F
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041

© 2000-2025, MetaQuotes Ltd.

식별자	값
CL_DEVICE_PARENT_DEVICE	0x1042
CL_DEVICE_PARTITION_MAX_SUB_DEVICES	0x1043
CL_DEVICE_PARTITION_PROPERTIES	0x1044
CL_DEVICE_PARTITION_AFFINITY_DOMAIN	0x1045
CL_DEVICE_PARTITION_TYPE	0x1046
CL_DEVICE_REFERENCE_COUNT	0x1047
CL_DEVICE_PREFERRED_INTEROP_USER_SYNC	0x1048
CL_DEVICE_PRINTF_BUFFER_SIZE	0x1049
CL_DEVICE_IMAGE_PITCH_ALIGNMENT	0x104A
CL_DEVICE_IMAGE_BASE_ADDRESS_ALIGNMENT	0x104B

예:

```
void OnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERROR in CLContextCreate");
      string device;
      CLGetInfoString(clCtx,CL_DEVICE_NAME,device);
      Print(i,": ",device);
      uchar data[1024];
      uint size;
      CLGetDeviceInfo(clCtx,CL_DEVICE_VENDOR,data,size);
      Print("size = ",size);
      string str=CharArrayToString(data);
      Print(str);
     }
  }
//--- Experts 저널의 항목 예제
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      2: Advanced Micro Devices, Inc.
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 32
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Tahiti
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      Intel(R) Corporation
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 21
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      1:        Intel(R) Core(TM) i7-377
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      NVIDIA Corporation
```

© 2000-2025, MetaQuotes Ltd.

```
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      size = 19
//  2013.07.24 10:50:48     opencl (EURUSD,H1)      0: GeForce GTX 580
```

© 2000-2025, MetaQuotes Ltd.

CLProgramCreate

소스 코드에서 OpenCL 프로그램을 생성합니다.

```
int  CLProgramCreate(
   int           context,     // OpenCL 컨텍스트 처리
   const string  source       // 소스 코드
   );
```

오버로드된 함수 버전은 OpenCL 프로그램을 생성하고 컴파일러 메시지를 전달된 문자열에 기록합니다.

```
int  CLProgramCreate(
   int           context,     // OpenCL 컨텍스트 처리
   const string  source,      // 소스 코드
   string       &build_log    // 컴파일 로그를 수신하기 위한 문자열
   );
```

Parameter

context

[in] OpenCL 컨텍스트 처리.

source

[in] OpenCL 프로그램의 소스 코드가 있는 문자열입니다.

&build_log

[in] OpenCL 컴파일러 메시지를 수신하는 문자열.

반환값

성공한 경우 OpenCL 개체에 대한 핸들. 오류 발생 시 -1이 반환됩니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

참고

현재 다음과 같은 오류 코드가 사용됩니다.

- ERR_OPENCL_INVALID_HANDLE - 컨텍스트에 대한 잘못된 핸들 OpenCL.
- ERR_INVALID_PARAMETER - 잘못된 문자열 매개 변수.
- ERR_NOT_ENOUGH_MEMORY - 메모리가 부족하여 작업을 완료할 수 없음.
- ERR_OPENCL_PROGRAM_CREATE - OpenCL의 내부 오류 또는 컴파일 오류.

일부 그래픽 카드에서는 기본적으로 double 유형 번호로 작업할 수 있습니다. 이로 인해 컴파일 오류 5105가 발생할 수 있습니다. double 유형 번호에 대한 지원을 활성화하려면 OpenCL 프로그램에 #pragma OPENCL EXTENSION cl_khr_fp64 : enable 명령을 추가하십시오. 그러나 그래픽 카드가 double을 지원하지 않는 경우 이 명령문을 활성화하는 것은 도움이 되지 않습니다.

예:

```
//+------------------------------------------------------------------+
//| OpenCL kernel                                                    |
//+------------------------------------------------------------------+
const string
cl_src=
```

© 2000-2025, MetaQuotes Ltd.

```
        //--- 기본적으로 일부 GPU는 double을 지원하지 않습니다
        //--- cl_khr_fp64 명령어는 복수로 작업을 활성화하는 데 사용됩니다
        "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
        //--- OpenCL 커널 함수
        "__kernel void Test_GPU(__global double *data,      \r\n"
        "                       const    int N,             \r\n"
        "                       const    int total_arrays)  \r\n"
        "  {                                                 \r\n"
        "   uint kernel_index=get_global_id(0);              \r\n"
        "   if (kernel_index>total_arrays) return;           \r\n"
        "   uint local_start_offset=kernel_index*N;          \r\n"
        "   for(int i=0; i<N; i++)                           \r\n"
        "     {                                              \r\n"
        "       data[i+local_start_offset] *= 2.0;           \r\n"
        "     }                                              \r\n"
        "  }                                                 \r\n";
//+------------------------------------------------------------------+
//| Test_CPU                                                         |
//+------------------------------------------------------------------+
bool Test_CPU(double &data[],const int N,const int id,const int total_arrays)
  {
//--- 배열 사이즈 체크
   if(ArraySize(data)==0) return(false);
//--- 배열 인덱스 체크
   if(id>total_arrays) return(false);
//--- 인덱스 ID를 사용하여 배열의 로컬 오프셋 계산
   int local_start_offset=id*N;
//--- 요소에 2를 곱함
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   return true;
  }
//---
#define ARRAY_SIZE   100  // 배열의 크기
#define TOTAL_ARRAYS 5    // 총 배열
//--- OpenCL 핸들
int cl_ctx;  // OpenCL 컨텍스트 핸들
int cl_prg;  // OpenCL 프로그램 핸들
int cl_krn;  // OpenCL 커널 핸들
int cl_mem;  // OpenCL 버퍼 핸들
//---
double DataArray1[]; // CPU 연산을 위한 데이터 배열
double DataArray2[]; // GPU 연산을 위한 데이터 배열
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
int OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- OpenCL 개체 초기화
//--- OpenCL 컨텍스트 생성
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL을 찾을 수 없습니다. Error=",GetLastError());
      return(1);
     }
//--- OpenCL 프로그램 생성
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
     {
      CLContextFree(cl_ctx);
      Print("OpenCL 프로그램 생성 실패. Error=",GetLastError());
      return(1);
     }
//--- OpenCL 커널 생성
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
     {
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL kernel 생성 실패. Error=",GetLastError());
      return(1);
     }
//--- OpenCL 버퍼 생성
   if((cl_mem=CLBufferCreate(cl_ctx,ARRAY_SIZE*TOTAL_ARRAYS*sizeof(double),CL_MEM_REAI
     {
      CLKernelFree(cl_krn);
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL 버퍼 생성 실패. Error=",GetLastError());
      return(1);
     }
//--- OpenCL 커널 상수 매개 변수 설정
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- 데이터 배열 준비
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- 배열에 데이터를 채웁니다
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- j 번째 배열에 대한 로컬 시작 오프셋 계산
      uint local_offset=j*ARRAY_SIZE;
      //--- 인덱스 j 로 배열 준비
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         //--- MathCos(i+j) 함수로 배열을 채웁니다;
         DataArray1[i+local_offset]=MathCos(i+j);
```

© 2000-2025, MetaQuotes Ltd.

```
         DataArray2[i+local_offset]=MathCos(i+j);
        }
      };
//--- CPU 연산 테스트
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- 인덱스 j를 사용하여 배열 계산
      Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
     }
//--- CLExecute 매개변수 준비
   uint  offset[]={0};
//--- 전 세계 작업 규모
   uint  work[]={TOTAL_ARRAYS};
//--- OpenCL 버퍼에 데이터 쓰기
   CLBufferWrite(cl_mem,DataArray2);
//--- OpenCL 커널을 실행
   CLExecute(cl_krn,1,offset,work);
//--- OpenCL 버퍼에서 데이터 읽기
   CLBufferRead(cl_mem,DataArray2);
//--- 총 에러
   double total_error=0;
//--- 결과를 비교하고 오차를 계산
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- j번째 배열에 대한 로컬 오프셋 계산
      uint local_offset=j*ARRAY_SIZE;
      //--- 결과를 비교
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         double v1=DataArray1[i+local_offset];
         double v2=DataArray2[i+local_offset];
         double delta=MathAbs(v2-v1);
         total_error+=delta;
         //--- 첫 번째 및 마지막 배열을 표시
         if((j==0) || (j==TOTAL_ARRAYS-1))
            PrintFormat("array %d of %d, element [%d]:  %f, %f, [error]=%f",j+1,TOTAL_
        }
     }
   PrintFormat("Total error: %f",total_error);
//--- OpenCL 개체 삭제
//--- free OpenCL 버퍼
   CLBufferFree(cl_mem);
//--- free OpenCL 커널
   CLKernelFree(cl_krn);
//--- free OpenCL 프로그램
   CLProgramFree(cl_prg);
//--- free OpenCL 컨텍스트
   CLContextFree(cl_ctx);
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

CLProgramFree

OpenCL 프로그램을 제거합니다.

```
void  CLProgramFree(
   int  program     // OpenCL 개체 처리
   );
```

Parameter

program

[in] OpenCL 개체 핸들.

반환값

없음. 내부 오류의 경우 _LastError의 값이 변경됩니다. 오류에 대해서는 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

CLKernelCreate

OpenCL 프로그램 커널을 생성하고 핸들을 반환합니다.

```
int  CLKernelCreate(
   int           program,        // OpenCL 개체 핸들
   const string  kernel_name     // Kernel 명
   );
```

Parameter

program

[in] OpenCL 프로그램의 개체에 대한 핸들.

kernel_name

[in] 실행이 시작되는 적절한 OpenCL 프로그램에서 커널 함수의 이름.

반환값

성공한 경우 OpenCL 개체에 대한 핸들. 오류 발생 시 -1이 반환됩니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

참고

현재 다음과 같은 오류 코드가 사용됩니다.

- ERR_OPENCL_INVALID_HANDLE - OpenCL 프로그램에 대한 핸들이 잘못됨.
- ERR_INVALID_PARAMETER - 문자열 매개변수가 잘못됨.
- ERR_OPENCL_TOO_LONG_KERNEL_NAME - 커널명이 127자 이상을 포함하고 있음.
- ERR_OPENCL_KERNEL_CREATE - OpenCL 개체를 생성하는 동안 내부 오류 발생.

© 2000-2025, MetaQuotes Ltd.

CLKernelFree

OpenCL 시작 함수를 제거합니다.

```
void  CLKernelFree(
   int  kernel     // OpenCL 프로그램의 커널에 대한 핸들
   );
```

Parameter

kernel_name

[in] 커널 개체의 핸들.

반환값

없음. 내부 오류의 경우 _LastError의 값이 변경됩니다. 오류에 대해서는 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArg

OpenCL 함수의 매개 변수를 설정합니다.

```
bool  CLSetKernelArg(
   int   kernel,        // OpenCL 프로그램의 커널에 대한 핸들
   uint  arg_index,     // OpenCL 함수의 인수 수
   void  arg_value      // 소스 코드
   );
```

Parameter

kernel

[in] OpenCL 프로그램의 커널에 대한 핸들.

arg_index

[in] 함수 인수의 숫자, 번호 매기는 0으로 시작함.

arg_value

[in] 함수 인수의 값.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

참고

현재 다음과 같은 오류 코드가 사용됩니다.

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE - OpenCL 커널에 대한 잘못된 핸들.
- ERR_OPENCL_SET_KERNEL_PARAMETER - OpenCL 내부 오류.

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArgMem

OpenCL 버퍼를 OpenCL 기능의 매개 변수로 설정합니다.

```
bool  CLSetKernelArgMem(
   int   kernel,           // OpenCL 프로그램의 커널에 대한 핸들
   uint  arg_index,        // OpenCL 기능의 인수의 개수입니다.
   int   cl_mem_handle     // 핸들을 OpenCL 버퍼에 연결
   );
```

Parameter

kernel

[in] OpenCL 프로그램의 커널에 대한 핸들.

arg_index

[in] 함수 인수의 숫자, 번호 매기는 0으로 시작함.

cl_mem_handle

[in] OpenCL 버퍼에 대한 핸들.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

CLSetKernelArgMemLocal

로컬 버퍼를 커널 함수의 인수로 설정합니다.

```
bool  CLSetKernelArgMemLocal(
   int    kernel,           // OpenCL 프로그램의 커널에 대한 핸들링
   uint   arg_index,        // OpenCL 함수 인수 수
   ulong  local_mem_size    // 버퍼 사이즈
   );
```

Parameter

kernel

[in] OpenCL 프로그램의 커널에 대한 핸들.

arg_index

[in] 함수 인수의 숫자, 번호 매기는 0으로 시작함.

local_mem_size

[in] 버퍼 크기(바이트).

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

CLBufferCreate

OpenCL 버퍼를 생성하고 핸들을 반환합니다.

```
int  CLBufferCreate(
OpenCL 컨텍스트 처리   int   context,     // Handle to an OpenCL context
   uint  size,         // 버퍼 사이즈
   uint  flags         // OpenCL 버퍼의 속성을 지정하는 플래그 조합
   );
```

Parameter

context

[in] OpenCL 컨텍스트에 대한 핸들.

size

[in] 버퍼 크기(바이트).

flags

[in] 플래그 조합을 사용하여 설정한 버퍼 속성: CL_MEM_READ_WRITE, CL_MEM_WRITE_ONLY, CL_MEM_READ_ONLY, CL_MEM_ALLOC_HOST_PTR.

반환값

성공한 경우의 OpenCL 버퍼에 대한 핸들. 오류 발생 시 -1이 반환됩니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

참고

현재 다음과 같은 오류 코드가 사용됩니다.

- ERR_OPENCL_INVALID_HANDLE - OpenCL 컨텍스트에 대한 핸들이 잘못됨.
- ERR_NOT_ENOUGH_MEMORY - 메모리 불충분.
- ERR_OPENCL_BUFFER_CREATE - 버퍼 생성 내부 오류.

© 2000-2025, MetaQuotes Ltd.

CLBufferFree

OpenCL 버퍼를 삭제합니다.

```
void  CLBufferFree(
   int   buffer     // OpenCL 버퍼에 대한 핸들링
   );
```

Parameter

buffer

[in] OpenCL 버퍼에 대한 핸들.

반환값

없음. 내부 오류의 경우 _LastError의 값이 변경됩니다. 오류에 대해서는 GetLastError() 함수를 사용하십시오.

© 2000-2025, MetaQuotes Ltd.

CLBufferWrite

OpenCL 버퍼에 쓰고 작성된 요소 수를 반환합니다.

```
uint  CLBufferWrite(
   int           buffer,                  // OpenCL 버퍼에 대한 핸들
   const void&   data[],                  // 값들의 배열
   uint          buffer_offset=0,         // OpenCL 버퍼의 오프셋(바이트), 기본적으로 0
   uint          data_offset=0,           // 요소 배열의 오프셋, 기본적으로 0
   uint          data_count=WHOLE_ARRAY   // 기본적으로 전체 배열을 쓰기 위한 배열의 값 수
   );
```

행렬과 벡터를 처리하기 위한 버전이 있습니다.

행렬의 값을 버퍼에 쓰고 성공하면 true를 반환합니다.

```
uint  CLBufferWrite(
   int           buffer,                  // OpenCL 버퍼에 대한 핸들
   uint          buffer_offset,           // OpenCL 버퍼의 오프셋(바이트)
   matrix<T>     &mat                     // 버퍼에 쓰기 위한 값 행렬
   );
```

벡터의 값을 버퍼에 쓰고 성공하면 true를 반환합니다.

```
uint  CLBufferWrite(
   int           buffer,                  // OpenCL 버퍼에 대한 핸들
   uint          buffer_offset,           // OpenCL 버퍼의 오프셋(바이트)
   vector<T>     &vec                     // 버퍼에 쓰기 위한 값 벡터
   );
```

Parameter

buffer

[in] OpenCL 버퍼의 핸들.

data[]

[in] OpenCL 버퍼에 기록해야 하는 값의 배열. 참조에 의해 통과됨

buffer_offset

[in] 쓰기가 시작되는 OpenCL 버퍼의 오프셋(바이트). 기본적으로 쓰기의 시작은 버퍼의 맨 처음입니다.

data_offset

[in] OpenCL 버퍼에 쓰여지는 배열의 값을 시작하는 첫 번째 배열 요소의 인덱스. 기본적으로 배열의 맨 처음부터 값을 가져옵니다.

data_count

[in] 기록해야 할 값 수. 기본적으로 배열의 모든 값.

mat

[out] 버퍼에서 데이터를 읽기 위한 행렬은 다음 세 가지 유형 중 하나일 수 있습니다 – matrix, matrixf 또는 matrixc.

© 2000-2025, MetaQuotes Ltd.

vec

[out] 버퍼에서 데이터를 읽기 위한 벡터는 vector, vectorf 또는 vectorc의 세 가지 유형 중 하나일 수 있습니다.

반환값

작성된 요소 수. 오류 발생 시 0이 반환됩니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

true 행렬 또는 벡터가 성공적으로 처리될 경우 그렇지 않을 경우 false.

참고

1차원 배열의 경우, OpenCL 버퍼에 쓰기 위한 데이터 읽기가 시작되는 요소의 수는 AS_SERIES 플래그를 고려하여 계산됩니다.

2개 이상의 차원으로 구성된 배열은 1차원으로 표시됩니다. 이 경우 *data_offset*은 프레젠테이션에서 건너뛸 요소의 수이지 첫 번째 차원의 요소 수가 아닙니다.

MatMul 메서드를 사용한 행렬 곱셈 및 OpenCL의 병렬 컴퓨팅의 예

```
#define M       3000      // 첫 번째 행렬의 행 수
#define K       2000      // 첫 번째 행렬의 열 수는 두 번째 행렬의 행 수와 같습니다.
#define N       3000      // 두 번째 행렬의 열 수

/+------------------------------------------------------------------+
const string clSrc=
  "#define N     "+IntegerToString(N)+"                              \r\n"
  "#define K     "+IntegerToString(K)+"                              \r\n"
  "                                                                 \r\n"
  "__kernel void matricesMul( __global float *in1,                  \r\n"
  "                           __global float *in2,                  \r\n"
  "                           __global float *out  )                \r\n"
  "{                                                                \r\n"
  "  int m = get_global_id( 0 );                                    \r\n"
  "  int n = get_global_id( 1 );                                    \r\n"
  "  float sum = 0.0;                                               \r\n"
  "  for( int k = 0; k < K; k ++ )                                  \r\n"
  "     sum += in1[ m * K + k ] * in2[ k * N + n ];                 \r\n"
  "  out[ m * N + n ] = sum;                                        \r\n"
  "}                                                                \r\n";
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
void OnStart()
 {
//--- 랜덤 넘버 생성기 초기화
  MathSrand((int)TimeCurrent());
//--- 랜덤 값으로 주어진 크기의 행렬 채우기
  matrixf mat1(M, K, MatrixRandom) ;    // first matrix
```

© 2000-2025, MetaQuotes Ltd.

```
  matrixf mat2(K, N, MatrixRandom);      // second matrix

//--- 네이티브 메서드를 사용하여 행렬의 곱을 계산
  uint start=GetTickCount();
  matrixf matrix_naive=matrixf::Zeros(M, N);// 두 행렬을 곱한 결과가 여기에 설정됩니다.
  for(int m=0; m<M; m++)
    for(int k=0; k<K; k++)
      for(int n=0; n<N; n++)
        matrix_naive[m][n]+=mat1[m][k]*mat2[k][n];
  uint time_naive=GetTickCount()-start;

//--- MatMul을 통해 행렬의 곱을 계산
  start=GetTickCount();
  matrixf matrix_matmul=mat1.MatMul(mat2);
  uint time_matmul=GetTickCount()-start;

//--- OpenCL에서 행렬의 곱 계산
  matrixf matrix_opencl=matrixf::Zeros(M, N);
  int cl_ctx;              // 컨텍스트 핸들
  if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL not found, leaving");
    return;
   }
  int cl_prg;              // 프로그램 핸들
  int cl_krn;              // 커널 핸들
  int cl_mem_in1;          // 첫번째(입력) 버퍼 핸들
  int cl_mem_in2;          // 두번째(입력) 버퍼 핸들e
  int cl_mem_out;          // 세번째(입력) 버퍼 핸들
//--- 프로그램과 커널 생성
  cl_prg = CLProgramCreate(cl_ctx, clSrc);
  cl_krn = CLKernelCreate(cl_prg, "matricesMul");
//--- 3개의 행렬에 대해 3개의 버퍼를 모두 생성
  cl_mem_in1=CLBufferCreate(cl_ctx, M*K*sizeof(float), CL_MEM_READ_WRITE);
  cl_mem_in2=CLBufferCreate(cl_ctx, K*N*sizeof(float), CL_MEM_READ_WRITE);
//--- 세번째 행렬 - 출력
  cl_mem_out=CLBufferCreate(cl_ctx, M*N*sizeof(float), CL_MEM_READ_WRITE);
//--- 커널 인수 설정
  CLSetKernelArgMem(cl_krn, 0, cl_mem_in1);
  CLSetKernelArgMem(cl_krn, 1, cl_mem_in2);
  CLSetKernelArgMem(cl_krn, 2, cl_mem_out);
//--- 장치 버퍼에 행렬 쓰기
  CLBufferWrite(cl_mem_in1, 0, mat1);
  CLBufferWrite(cl_mem_in2, 0, mat2);
  CLBufferWrite(cl_mem_out, 0, matrix_opencl);
//--- OpenCL 코드 실행 시간 시작
  start=GetTickCount();
//--- 태스크 작업 영역의 매개변수를 설정하고 OpenCL 프로그램을 실행
  uint  offs[2] = {0, 0};
```

© 2000-2025, MetaQuotes Ltd.

```
  uint works[2] = {M, N};
  start=GetTickCount();
  bool ex=CLExecute(cl_krn, 2, offs, works);
//--- 결과를 행렬로 계산
  if(CLBufferRead(cl_mem_out, 0, matrix_opencl))
    PrintFormat("[%d x %d] matrix read: ", matrix_opencl.Rows(), matrix_opencl.Cols())
   else
      Print("CLBufferRead(cl_mem_out, 0, matrix_opencl failed. Error ",GetLastError())
  uint time_opencl=GetTickCount()-start;
  Print("Compare calculation time using each method");
  PrintFormat("Naive product time = %d ms",time_naive);
  PrintFormat("MatMul product time = %d ms",time_matmul);
  PrintFormat("OpenCl product time = %d ms",time_opencl);
//--- 모든 OpenCL 컨텍스트 해제
  CLFreeAll(cl_ctx, cl_prg, cl_krn, cl_mem_in1, cl_mem_in2, cl_mem_out);

//--- 얻은 모든 결과 행렬을 서로 비교
  Print("How many discrepancy errors are there between result matrices?");
  ulong errors=matrix_naive.Compare(matrix_matmul,(float)1e-12);
  Print("matrix_direct.Compare(matrix_matmul,1e-12)=",errors);
  errors=matrix_matmul.Compare(matrix_opencl,float(1e-12));
  Print("matrix_matmul.Compare(matrix_opencl,1e-12)=",errors);
/*
  결과:

   [3000 x 3000] matrix read:
  각 메서드별 계산 시간 비교
   Naive product time = 54750 ms
   MatMul product time = 4578 ms
   OpenCl product time = 922 ms
  결과 행렬 사이에 얼마나 많은 불일치 오류가 있을까요?
   matrix_direct.Compare(matrix_matmul,1e-12)=0
   matrix_matmul.Compare(matrix_opencl,1e-12)=0
*/
 }
/+------------------------------------------------------------------+
//| 행렬을 임의의 값으로 채웁니다.                                   |
/+------------------------------------------------------------------+
void MatrixRandom(matrixf& m)
 {
  for(ulong r=0; r<m.Rows(); r++)
   {
    for(ulong c=0; c<m.Cols(); c++)
     {
      m[r][c]=(float)((MathRand()-16383.5)/32767.);
     }
   }
 }
/+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 모든 OpenCL 컨텍스트 해제                                                |
/+------------------------------------------------------------------+
void CLFreeAll(int cl_ctx, int cl_prg, int cl_krn,
               int cl_mem_in1, int cl_mem_in2, int cl_mem_out)
 {
//--- OpenCL에서 생성한 모든 컨텍스트를 역순으로 삭제
  CLBufferFree(cl_mem_in1);
  CLBufferFree(cl_mem_in2);
  CLBufferFree(cl_mem_out);
  CLKernelFree(cl_krn);
  CLProgramFree(cl_prg);
  CLContextFree(cl_ctx);
 }
```

© 2000-2025, MetaQuotes Ltd.

CLBufferRead

OpenCL 버퍼를 배열로 읽고 읽기 요소 수를 반환합니다.

```
uint  CLBufferRead(
   int           buffer,                    // OpenCL 버퍼에 대한 핸들
   const void&   data[],                    // 값들의 배열
   uint          buffer_offset=0,           // OpenCL 버퍼의 오프셋(바이트), 기본적으로 0
   uint          data_offset=0,             // 요소 배열의 오프셋, 기본적으로 0
   uint          data_count=WHOLE_ARRAY     // 기본적으로 전체 버퍼를 읽기 위한 버퍼의 값의
   );
```

행렬과 벡터를 처리하기 위한 버전이 있습니다.

OpenCL 버퍼를 매트릭스로 읽고 성공하면 true를 반환합니다.

```
uint  CLBufferRead(
   int           buffer,                    // OpenCL 버퍼에 대한 핸들
   uint          buffer_offset,             // OpenCL 버퍼의 오프셋(바이트)
   const matrxi& mat,                       // 버퍼에서 값을 수신하기 위한 행렬
   ulong         rows=-1,                   // 행렬의 행 수
   ulong         cols=-1                    // 행렬의 열 수
   );
```

OpenCL 버퍼를 벡터로 읽고 성공하면 true를 반환합니다.

```
uint  CLBufferRead(
   int           buffer,                    // OpenCL 버퍼에 대한 핸들
   uint          buffer_offset,             // OpenCL 버퍼의 오프셋(바이트)
   const vector& vec,                       // 버퍼에서 값을 수신하기 위한 벡터
   ulong         size-1,                    // 벡터 길이
   );
```

Parameter

buffer

[in] OpenCL 버퍼의 핸들.

data[]

[in] OpenCL 버퍼에서 값을 수신하기 위한 배열. 참조에 의해 통과됨

buffer_offset

[in] OpenCL 버퍼의 오프셋(바이트)이며, 여기서 읽기가 시작됩니다. 기본적으로 읽기는 버퍼의 처음 시작부터 시작합니다.

data_offset

[in] OpenCL 버퍼의 값을 쓰기 위한 첫 번째 배열 요소의 인덱스. 기본적으로 읽기 값을 배열로 쓰는 작업은 제로 인덱스에서 시작됩니다.

data_count

[in] 읽어야 할 값의 수. 전체 OpenCL 버퍼는 기본적으로 읽힙니다.

mat

© 2000-2025, MetaQuotes Ltd.

[out] 버퍼에서 데이터를 읽기 위한 행렬은 다음 세 가지 유형 중 하나일 수 있습니다 – matrix, matrixf 또는 matrixc.

vec

[out] 버퍼에서 데이터를 읽기 위한 벡터는 vector, vectorf 또는 vectorc의 세 가지 유형 중 하나일 수 있습니다.

rows=-1

[in] 매개변수가 지정되면 *cols* 매개변수도 지정되어야 합니다. 새 행렬의 차원을 지정하지 않으면 현재 차원이 사용됩니다. 값이 -1이면 행 수가 변경되지 않습니다.

cols=-1

[in] 매개변수를 지정하지 않으면 *rows* 매개변수도 건너뛰어야 합니다. 매트릭스는 규칙을 따릅니다: 두 매개변수가 모두 지정되거나 지정되지 않으면 오류가 발생합니다. 두 매개변수(*rows* 및 *cols*)가 모두 지정되면 행렬 크기가 변경됩니다. -1의 경우 컬럼의 개수는 변하지 않습니다.

size=-1

[in] 매개변수가 지정되지 않았거나 값이 -1이면 벡터 길이가 변경되지 않습니다.

반환값

읽기 요소의 수. 오류 발생 시 0이 반환됩니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

true 행렬 또는 벡터가 성공적으로 처리될 경우 그렇지 않을 경우 false.

참고

1차원 배열의 경우, OpenCL 버퍼에 데이터를 쓰기 시작하는 요소의 수는 AS_SERIES 플래그를 고려하여 계산됩니다.

2개 이상의 차원으로 구성된 배열은 1차원으로 표시됩니다. 이 경우 *data_offset*은 프레젠테이션에서 건너뛸 요소의 수이지 첫 번째 차원의 요소 수가 아닙니다.

방정식을 사용하여 PI 계산 예:

$$\pi = \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N-1}\frac{4}{1+\left(\frac{2k+1}{2N}\right)^2} = 16\lim_{N\to\infty}N\sum_{k=0}^{N-1}\frac{1}{4N^2+(2k+1)^2}$$

```
#define  _num_steps        1000000000
#define  _divisor          40000
#define  _step             1.0 / _num_steps
#define  _intrnCnt         _num_steps / _divisor

/+------------------------------------------------------------------+
//|                                                                  |
/+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
string D2S(double arg, int digits) { return DoubleToString(arg, digits); }
string I2S(int arg)                { return IntegerToString(arg); }

//--- OpenCL 프로그램 코드
const string clSource=
  "#define _step "+D2S(_step, 12)+"                    \r\n"
  "#define _intrnCnt "+I2S(_intrnCnt)+"                \r\n"
  "                                                    \r\n"
  "__kernel void Pi( __global double *out )            \r\n"
  "{                                                   \r\n"
  "  int i = get_global_id( 0 );                       \r\n"
  "  double partsum = 0.0;                             \r\n"
  "  double x = 0.0;                                   \r\n"
  "  long from = i * _intrnCnt;                        \r\n"
  "  long to = from + _intrnCnt;                       \r\n"
  "  for( long j = from; j < to; j ++ )                \r\n"
  "  {                                                 \r\n"
  "     x = ( j + 0.5 ) * _step;                       \r\n"
  "     partsum += 4.0 / ( 1. + x * x );               \r\n"
  "  }                                                 \r\n"
  "  out[ i ] = partsum;                               \r\n"
  "}                                                   \r\n";

/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
int OnStart()
 {
  Print("Pi Calculation: step = "+D2S(_step, 12)+"; _intrnCnt = "+I2S(_intrnCnt));
//--- OpenCL 컨텍스트 준비
  int clCtx;
  if((clCtx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL not found");
    return(-1);
   }
  int clPrg = CLProgramCreate(clCtx, clSource);
  int clKrn = CLKernelCreate(clPrg, "Pi");
  int clMem=CLBufferCreate(clCtx, _divisor*sizeof(double), CL_MEM_READ_WRITE);
  CLSetKernelArgMem(clKrn, 0, clMem);

  const uint offs[1]  = {0};
  const uint works[1] = {_divisor};
//--- OpenCL 프로그램 실행
  ulong start=GetMicrosecondCount();
  if(!CLExecute(clKrn, 1, offs, works))
   {
    Print("CLExecute(clKrn, 1, offs, works) failed! Error ", GetLastError());
    CLFreeAll(clMem, clKrn, clPrg, clCtx);
```

© 2000-2025, MetaQuotes Ltd.

```
     return(-1);
    }
//--- OpenCL 기기로부터 결과
  vector buffer(_divisor);
  if(!CLBufferRead(clMem, 0, buffer))
    {
     Print("CLBufferRead(clMem, 0, buffer) failed! Error ", GetLastError());
     CLFreeAll(clMem, clKrn, clPrg, clCtx);
     return(-1);
    }
//--- Pi 를 계산하기 위해 모든 값을 더함
  double Pi=buffer.Sum()*_step;

  double time=(GetMicrosecondCount()-start)/1000.;
  Print("OpenCL: Pi calculated for "+D2S(time, 2)+" ms");
  Print("Pi = "+DoubleToString(Pi, 12));
//--- 메모리 비우기
  CLFreeAll(clMem, clKrn, clPrg, clCtx);
//--- 성공
  return(0);
 }
  /*
  Pi Calculation: step = 0.000000001000; _intrnCnt = 25000
  OpenCL: GPU device 'Ellesmere' selected
  OpenCL: Pi calculated for 99.98 ms
  Pi = 3.141592653590
  */
/+------------------------------------------------------------------+
//| 메모리를 확보하는 보조 루틴                                      |
/+------------------------------------------------------------------+
void CLFreeAll(const int clMem, const int clKrn, const int clPrg, const int clCtx)
 {
  CLBufferFree(clMem);
  CLKernelFree(clKrn);
  CLProgramFree(clPrg);
  CLContextFree(clCtx);
 }
```

© 2000-2025, MetaQuotes Ltd.

CLExecute

이 함수는 OpenCL 프로그램을 실행합니다. 이 함수에는 세 가지 버전이 있습니다:

1. 하나의 커널을 사용하여 커널 함수 시작

```
bool  CLExecute(
   int          kernel                  // OpenCL 프로그램의 커널에 대한 핸들
   );
```

2. 작업 공간 설명이 포함된 여러 커널 복사본(OpenCL 함수) 시작

```
bool  CLExecute(
   int          kernel,                 // OpenCL 프로그램의 커널에 대한 핸들
   uint         work_dim,               // 작업 공간의 규모
   const uint&  global_work_offset[],   // 작업 공간의 초기 오프셋
   const uint&  global_work_size[]      // 총 작업 수
   );
```

3. 작업 공간 설명 및 그룹 로컬 작업 하위 집합의 크기 지정을 사용하여 여러 커널 복사본(OpenCL 함수) 시작

```
bool  CLExecute(
   int          kernel,                 // OpenCL 프로그램의 커널에 대한 핸들
   uint         work_dim,               // 작업 공간의 규모
   const uint&  global_work_offset[],   // 작업 공간의 초기 오프셋
   const uint&  global_work_size[],     // 총 작업 수
   const uint&  local_work_size[]       // 로컬 그룹의 작업 수
   );
```

Parameter

kernel

[in] OpenCL 커널에 대한 핸들.

work_dim

[in] 작업 공간의 규모.

global_work_offset[]

[in] 작업 공간의 초기 오프셋.

global_work_size[]

[in] 작업 하위 집합의 크기.

local_work_size[]

[in] 그룹의 로컬 작업 하위 집합의 크기.

반환값

성공하면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용은 GetLastError() 함수를 사용하십시오.

참고

© 2000-2025, MetaQuotes Ltd.

다음 예에서 매개 변수의 사용을 고려하십시오.

- *work_dim*은 작업을 설명하는 *work_items[]* 배열의 크기를 지정합니다. *work_dim*=3일 경우, 3차원 배열 *work_items[N1, N2, N3]*이 사용됩니다.
- *global_work_size[]*에는 *work_items[]* 어레이 크기를 설정하는 값이 포함되어 있습니다. work_dim=3인 경우, *global_work_size[3]* 어레이는 {40, 100, 320}일 수 있습니다. 그러면 *work_items[40, 100, 320]*을 보유하게 됩니다. 따라서 총 작업 수는 40 x 100 x 320 = 1 280 000개가 됩니다.
- *local_work_size[]*은 OpenCL 프로그램의 지정된 커널에 의해 실행될 작업의 하위 집합을 설정합니다. 그것의 크기는 *work_items[]* 크기와 같으며, 분할에서 나머지의 손실 없이 공통 작업 부분 집합을 더 작은 하위 집합으로 분할할 수 있습니다. 실제로 *local_work_size[]* 배열의 크기를 선택해야 *work_items[]* 글로벌 작업 세트가 더 작은 하위 집합으로 분할됩니다. 이 예에서는 *local_work_size[3]={10, 10, 10}*이(가) 정상입니다. *work_items[40, 100, 320]*은(는) *local_items[10, 10, 10]* 배열에서 수집될 수 있기 때문입니다.

© 2000-2025, MetaQuotes Ltd.

CLExecutionStatus

OpenCL 프로그램 실행 상태를 반환합니다.

```
int  CLExecutionStatus(
   int   kernel            // OpenCL 프로그램의 커널에 대한 핸들링
   );
```

Parameter

kernel

[in] OpenCL 프로그램의 커널에 대한 핸들.

반환값

OpenCL 프로그램 상태를 반환합니다. 값은 다음 중 하나일 수 있습니다.

- CL_COMPLETE=0 - 프로그램 완료,
- CL_RUNNING=1 - 동작중,
- CL_SUBMITTED=2 - 실행을 위해 제출됨,
- CL_QUEUED=3 - 대기중,
- -1 (마이너스 1) - CLExecutionStatus()를 실행하는 동안 오류 발생.

© 2000-2025, MetaQuotes Ltd.

데이터베이스로 작업

데이터베이스 작업을 위한 기능은 널리 사용되고 사용이 간편한 SQLite 엔진을 적용합니다. 이 엔진의 편리한 기능은 전체 데이터베이스가 사용자 PC 하드 디스크의 단일 파일에 있다는 것입니다.

이 기능을 사용하면 테이블을 편리하게 생성하고, 여기에 데이터를 추가하고 수정 및 샘플링을 단순 SQL 요청을 사용하여 수행할 수 있습니다:

- 모든 형식에서 거래 내역 및 시세 수신,
- 최적화 및 테스트 결과 저장,
- 다른 분석 패키지와 데이터 준비 및 교환,
- MQL5 애플리케이션 설정 및 상태를 저장.

데이터베이스 작업 기능을 사용하면 가장 반복적인 대용량 데이터 배열 처리 작업을 SQL 요청으로 대체할 수 있으므로 복잡한 루프 및 비교를 프로그래밍 하는 대신 DatabaseExecute/DatabasePrepare 호출을 사용할 수 있습니다. DatabaseReadBind 기능을 사용하여 조회 결과를 미리 작성된 구조로 손쉽게 얻을 수 있습니다. 이 기능을 통해 단일 통화 내에서 모든 기록 필드를 한 번에 읽을 수 있습니다.

읽기, 쓰기 및 수정을 가속화하기 위해 DATABASE_OPEN_MEMORY 플래그를 사용하여 RAM에서 데이터베이스를 열거나 생성할 수 있으며, 이러한 데이터베이스는 특정 애플리케이션에서만 사용할 수 있고 공유되지 않습니다. 하드 디스크에 위치한 데이터베이스로 작업할 때는 DatabaseTransactionBegin/DatabaseTransactionCommit/DatabaseTransactionRollback을 사용하여 대량 데이터 삽입/변경 사항을 트랜잭션으로 묶어야 합니다. 이는 프로세스를 수백 번 가속화됩니다.

이 기능을 사용하려면 SQLite: MQL5의 SQL 데이터베이스 네이티브 처리 문서를 참조하십시오.

기능	동작
DatabaseOpen	지정된 파일에 데이터베이스를 열거나 생성
DatabaseClose	데이터베이스를 닫기
DatabaseImport	파일에서 테이블로 데이터 가져오기
DatabaseExport	테이블 또는 SQL 요청 실행 결과를 CSV 파일로 내보내기
DatabasePrint	SQL 요청 실행 결과를 엑스퍼트 저널에 인쇄
DatabaseTableExists	데이터베이스에 테이블이 있는지 확인
DatabaseExecute	지정된 데이터베이스에 대한 요청을 실행
DatabasePrepare	DatabaseRead()를 사용하여 실행할 수 있는 요청 핸들을 생성
DatabaseReset	DatabasePrepare()를 호출한 후와 같이 요청을 재설정
DatabaseBind	요청에 매개변수 값을 설정
DatabaseBindArray	배열을 매개변수 값을 설정
DatabaseRead	요청 결과로 다음 항목으로 이동
DatabaseReadBind	다음 기록으로 이동하여 기록에서 구조물로 데이터를 읽기
DatabaseFinalize	DatabasePrepare()에서 생성된 요청을 제거

© 2000-2025, MetaQuotes Ltd.

기능	동작
DatabaseTransactionBegin	트랜잭션 실행 시작
DatabaseTransactionCommit	트랜잭션 실행 완료
DatabaseTransactionRollback	트랜잭션을 롤백
DatabaseColumnsCount	요청의 필드 수를 가져오기
DatabaseColumnName	인덱스별로 필드 이름을 가져오기
DatabaseColumnType	인덱스별 필드 유형을 가져오기
DatabaseColumnSize	필드 크기(바이트)를 가져오기
DatabaseColumnText	현재 기록에서 문자열로 필드 값을 가져오기
DatabaseColumnInteger	현재 기록에서 int 유형 값을 가져오기
DatabaseColumnLong	현재 기록에서 long 유형 값을 가져오기
DatabaseColumnDouble	현재 기록에서 double 유형 값을 가져오기
DatabaseColumnBlob	현재 기록에서 배열로 필드 값을 가져오기

© 2000-2025, MetaQuotes Ltd.

DatabaseOpen

지정된 파일에 데이터베이스를 열거나 생성.

```
int  DatabaseOpen(
   string  filename,       // 파일 이름
   uint    flags           // 플래그의 조합
   );
```

매개변수

filename

[in] "MQL5₩Files" 폴더에 상대적인 파일 이름.

flags

[in] ENUM_DATABASE_OPEN_FLAGS 열거의 플래그 조합.

반환 값

성공적으로 실행되면 함수는 데이터베이스 핸들을 반환하고 이 핸들을 데이터베이스에 액세스하는 데 사용합니다. 그렇지 않으면 INVALID_HANDLE을 반환합니다. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INTERNAL_ERROR (4001) - 심각한 런타임 오류;
- ERR_WRONG_INTERNAL_PARAMETER (4002) - "MQL5₩Files" 폴더에 액세스 하는 도중 내부 오류;
- ERR_INVALID_PARAMETER (4003) - 데이터베이스 파일의 경로에 빈 문자열이 포함되어 있거나 호환되지 않는 플래그 조합이 설정;
- ERR_NOT_ENOUGH_MEMORY (4004) - 충분하지 않은 메모리;
- ERR_WRONG_FILENAME (5002) - 잘못된 데이터베이스 파일 이름;
- ERR_TOO_LONG_FILENAME (5003) - 데이터베이스 파일의 절대 경로가 최대 길이를 초과;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - 허용되는 최대 데이터베이스 개체 수를 초과;
- ERR_DATABASE_CONNECT (5123) - 데이터베이스 연결 오류;
- ERR_DATABASE_MISUSE (5621) - 잘못된 SQLite 라이브러리 사용.

참고

파일명 매개변수에 NULL 또는 빈 문자열 ""이 있으면 디스크에 임시 파일이 생성됩니다. 데이터베이스 연결을 닫은 후 자동으로 삭제됩니다.

파일명 매개변수가 ":memory:"인 경우 데이터베이스는 메모리에 생성되고 연결이 닫히면 자동으로 삭제됩니다.

플래그 매개변수에 DATABASE_OPEN_READONLY 또는 DATABASE_OPEN_READWRITE 플래그가 없는 경우 DATABASE_OPEN_READWRITE 플래그가 사용됩니다.

파일 확장명을 지정하지 않으면 ".sqlite"가 사용됩니다.

ENUM_DATABASE_OPEN_FLAGS

ID	설명
DATABASE_OPEN_READONLY	읽기 전용
DATABASE_OPEN_READWRITE	읽기 및 쓰기 위한 열기

© 2000-2025, MetaQuotes Ltd.

ID	설명
DATABASE_OPEN_CREATE	필요한 경우 디스크에 파일 생성
DATABASE_OPEN_MEMORY	RAM에 데이터베이스 생성
DATABASE_OPEN_COMMON	파일은 모든 터미널 공통 폴더에 있습니다

추가 참조

DatabaseClose

© 2000-2025, MetaQuotes Ltd.

DatabaseClose

Closes a database.

```
void  DatabaseClose(
   int  database      // DatabaseOpen에서 수신된 데이터베이스 핸들
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

반환 값

없음.

참고

DatabaseClose를 호출하면, 데이터베이스에 대한 모든요청 처리기가 자동으로 제거되고 유효하지 않게 됩니다.

핸들이 잘못된 경우 함수는 ERR_DATABASE_INVALID_HANDLE 오류를 설정합니다. GetLastError()를 사용하여 오류를 확인할 수 있습니다.

추가 참조

DatabaseOpen, DatabasePrepare

© 2000-2025, MetaQuotes Ltd.

DatabaseImport

데이터를 파일에서 테이블로 가져오기.

```
long  DatabaseImport(
   int           database,          // DatabaseOpen에서 수신된 데이터베이스 핸들
   const string  table,             // 데이터를 삽입할 테이블의 이름
   const string  filename,          // 데이터를 가져올 파일의 이름
   uint          flags,             // 플래그의 조합
   const string  separator,         // 데이터 분리자
   ulong         skip_rows,         // 생략할 초기 문자열 수
   const string  skip_comments      // 설명 정의 문자 문자열
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

table

[in] 파일의 데이터를 추가할 테이블의 이름.

filename

[in] 데이터를 읽기 위한 CSV 파일 또는 ZIP 아카이브. 이름은 하위 디렉토리를 포함할 수 있으며 MQL5₩Files 폴더를 기준으로 설정됩니다.

flags

[in] 플래그의 조합 ENUM_DATABASE_IMPORT_FLAGS 열거형에서.

separator

[in] CSV 파일의 데이터 분리자.

skip_rows

[in] 파일에서 데이터를 읽을 때 건너뛸 초기 문자열 수.

skip_comments

[in] 문자열을 코멘트로 지정하기 위한 문자열. 문자열 시작 부분에서 *skip_comments*의 문자가 감지되면 면 이러한 문자열은 주석으로 간주되며 가져오지 않습니다.

반환 값

가져온 문자열의 수를 반환하거나 오류 발생 시 -1을 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INVALID_PARAMETER (4003) – 테이블 이름이 지정되지 않음 (빈 문자열 또는 NULL);
- ERR_DATABASE_INTERNAL (5120) – 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) – 유효하지 않은 데이터베이스 핸들.

ENUM_DATABASE_IMPORT_FLAGS

© 2000-2025, MetaQuotes Ltd.

ID	설명
DATABASE_IMPORT_HEADER	첫 번째 줄에는 테이블 필드의 이름이 포함됩니다.
DATABASE_IMPORT_CRLF	CRLF (디폴트는 LF)는 문자열 나누기로 간주됩니다
DATABASE_IMPORT_APPEND	현재 테이블 끝에 데이터를 추가합니다
DATABASE_IMPORT_QUOTED_STRINGS	큰따옴표로 묶인 문자열 값
DATABASE_IMPORT_COMMON_FOLDER	파일은 모든 클라이언트 터미널의 ₩Terminal₩Common₩File의 공통 폴더에 저장됩니다.

DatabaseExport 예시 코드로 부터 생성된 파일에서 테이블을 읽는 예:

```
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
void OnStart()
 {
  string csv_filename;
//--- 클라이언트 터미널의 공통 폴더에서 다운로드할 텍스트 파일의 이름을 가져옵니다.
  string filenames[];
  if(FileSelectDialog("Select a CSV file to download a table", NULL,
                      "Text files (*.csv)|*.csv",
                      FSD_WRITE_FILE|FSD_COMMON_FOLDER, filenames, "data.csv")>0)
   {
 //--- 선택한 각 파일의 이름 표시
    if(ArraySize(filenames)==1)
      csv_filename=filenames[0];
    else
     {
      Print("Unknown error while selecting file. Error code ", GetLastError());
      return;
     }
   }
  else
   {
    Print("CSV file not selected");
    return;
   }
//--- 데이터 베이스 생성 혹은 열기
  string db_filename="test.sqlite";
  int db=DatabaseOpen(db_filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE);
//--- TEST 테이블이 존재하는지 확인
  if(DatabaseTableExists(db, "TEST"))
   {
    //--- TEST 테이블 제거
```

© 2000-2025, MetaQuotes Ltd.

```
    if(!DatabaseExecute(db, "DROP TABLE IF EXISTS TEST"))
     {
      Print("Failed to drop the TEST table with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
   }
 //--- 파일에서 TEST 테이블로 항목 가져오기
  long imported=DatabaseImport(db, "TEST", csv_filename, DATABASE_IMPORT_HEADER|DATABA
  if(imported>0)
   {
    Print(imported," lines imported in table TEST");
    DatabasePrint(db,"SELECT * FROM TEST",DATABASE_PRINT_NO_INDEX);
   }
  else
    {
     Print("DatabaseImport() failed. Error ",GetLastError());
    }
//--- 데이터 베이스 파일을 닫고 알리기
  DatabaseClose(db);
  PrintFormat("Database: %s closed", db_filename);
 }
```

참고

테이블 이름이 지정된 *테이블*이 없으면 자동으로 생성됩니다. 생성된 테이블의 이름 및 필드 유형은 파일 데이터를 기반으로 자동으로 정의됩니다.

추가 참조

DatabaseOpen, DatabasePrint

© 2000-2025, MetaQuotes Ltd.

DatabaseExport

테이블 또는 SQL 요청 실행 결과를 CSV 파일로 내보내기. 파일은 UTF-8 인코딩으로 생성됩니다.

```
long  DatabaseExport(
   int           database,          // DatabaseOpen에서 수신된 데이터베이스 핸들
   const string  table_or_sql,      // 테이블 이름 또는 SQL 요청
   const string  filename,          // 데이터 내보내기를 위한 CSV 파일 이름
   uint          flags,             // 플래그의 조합
   const string  separator          // CSV 파일의 데이터 분리자
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

table_or_sql

[in] 지정된 파일로 결과를 내보낼 테이블 또는 SQL 요청 텍스트의 이름.

filename

[in] 데이터 내보내기의 파일 이름. 경로는MQL5₩Files 폴더를 기준으로 설정됩니다.

flags

[in] 플래그의 조합 ENUM_DATABASE_EXPORT_FLAGS 열거형에서.

separator

[in] 데이터 분리자. NULL을 지정한 경우 '₩' 표 문자를 분리자로 사용. 빈 문자열 "" 은 유효한 분리자로 간주되지만, 가져온 CSV 파일은 테이블로 읽을 수 없으며, 문자열 집합으로 간주됩니다.

반환 값

오류가 발생할 경우 내보낸 항목 수 또는 음수 값을 반환합니다. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INTERNAL_ERROR (4001) – 심각한 런타임 오류;
- ERR_INVALID_PARAMETER (4003) – 데이터베이스 파일의 경로에 빈 문자열이 포함되어 있거나 호환되지 않는 플래그 조합이 설정;
- ERR_NOT_ENOUGH_MEMORY (4004) – 충분하지 않은 메모리;
- ERR_FUNCTION_NOT_ALLOWED(4014) – 지정된 파이프는 허용되지 않음;
- ERR_PROGRAM_STOPPED(4022) – 작업이 취소됨 (MQL 프로그램 중지됨);
- ERR_WRONG_FILENAME (5002) – 유효하지 않은 파일 이름;
- ERR_TOO_LONG_FILENAME (5003) – 파일의 절대 경로가 최대 길이를 초과;
- ERR_CANNOT_OPEN_FILE(5004) – 파일을 쓰기 위해 열 수 없음;
- ERR_FILE_WRITEERROR(5026) – 파일에 쓰기할 수 없음;
- ERR_DATABASE_INTERNAL (5120) – 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) – 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_QUERY_PREPARE(5125) – 생성 오류 요청;
- ERR_DATABASE_QUERY_NOT_READONLY – 읽기 전용 요청이 허용됨.

© 2000-2025, MetaQuotes Ltd.

참고

요청 결과를 내보내는 경우 SQL 요청은 "SELECT" 또는 "select"로 시작해야 합니다. 즉, SQL 요청에서 데이터 베이스 상태를 변경할 수 없으며, 그렇지 않으면 DatabaseExport()가 실패하고 오류가 발생합니다.

데이터베이스 문자열 값에는 변환 문자 ('₩' 또는 '₩₩')와 구분 기호 매개변수에 설정된 값 *분리자*가 포함될 수 있습니다. 이 경우 'flags' 매개변수에서 DATABASE_EXPORT_QUOTED_STRINGS 플래그를 사용해야 합니다. 이 플래그가 있으면 포시되는 모든 문자열이 double 시세로 묶입니다. 문자열에 double 시세가 포함된 경우, 두 double 시세로 교체됩니다.

ENUM_DATABASE_EXPORT_FLAGS

ID	설명
DATABASE_EXPORT_HEADER	첫 번째 문자열에 필드 이름 표시
DATABASE_EXPORT_INDEX	문자열 인덱스 표시
DATABASE_EXPORT_NO_BOM	파일 시작 부분에 BOM 마크를 삽입하지 마십시오(BOM은 기본적으로 삽입됩니다)
DATABASE_EXPORT_CRLF	문자열 나누기에 CRLF 사용(기본값은 LF)
DATABASE_EXPORT_APPEND	기존의 파일 끝에 데이터를 추가합니다(기본적으로 파일을 덮어씁니다). 파일이 없으면 생성됩니다.
DATABASE_EXPORT_QUOTED_STRINGS	큰따옴표로 묶인 문자열 값을 표시합니다.
DATABASE_EXPORT_COMMON_FOLDER	모든 클라이언트 터미널의 ₩Terminal₩Common₩Files의 공통 폴더에 CSV 파일이 생성됩니다.

예:

```
input int InpRates=100;
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
void OnStart()
 {
  MqlRates rates[];
//--- 바를 받기 전에 시작 시간을 기억
  ulong start=GetMicrosecondCount();
//--- 1시간 주기에서 최근 100개의 바를 요청
  if(CopyRates(Symbol(), PERIOD_H1, 1, InpRates, rates)<InpRates)
   {
    Print("CopyRates() failed,, Error ", GetLastError());
    return;
   }
```

© 2000-2025, MetaQuotes Ltd.

```
  else
    {
 //--- 수신된 바의 수와 수신하는 데 걸린 시간
    PrintFormat("%s: CopyRates received %d bars in %d ms ",
                _Symbol, ArraySize(rates), (GetMicrosecondCount()-start)/1000);
    }
//--- 데이터 베이스를 저장하기 위해 파일명 설정
  string filename=_Symbol+"_"+EnumToString(PERIOD_H1)+"_"+TimeToString(TimeCurrent())+
  StringReplace(filename, ":", "-"); // ":" character is not allowed in file names
//--- 공통 터미널 폴더에서 데이터베이스 열기/생성
  int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE|DATABASE_
  if(db==INVALID_HANDLE)
    {
    Print("Database: ", filename, " open failed with code ", GetLastError());
    return;
    }
  else
    Print("Database: ", filename, " opened successfully");

//--- RATES 테이블이 존재하는지 확인
  if(DatabaseTableExists(db, "RATES"))
    {
    //--- RATES 테이블 제거
    if(!DatabaseExecute(db, "DROP TABLE IF EXISTS RATES"))
      {
      Print("Failed to drop the RATES table with code ", GetLastError());
      DatabaseClose(db);
      return;
      }
    }
//--- RATES 테이블 생성
  if(!DatabaseExecute(db, "CREATE TABLE RATES("
                      "SYMBOL             CHAR(10),"
                      "TIME               INT NOT NULL,"
                      "OPEN               REAL,"
                      "HIGH               REAL,"
                      "LOW                REAL,"
                      "CLOSE              REAL,"
                      "TICK_VOLUME        INT,"
                      "SPREAD             INT,"
                      "REAL_VOLUME        INT);"))
    {
    Print("DB: ", filename, " create table RATES with code ", GetLastError());
    DatabaseClose(db);
    return;
    }
//--- RATES 테이블의 모든 필드 목록을 표시
  if(DatabasePrint(db, "PRAGMA TABLE_INFO(RATES)", 0)<0)
    {
```

© 2000-2025, MetaQuotes Ltd.

```
    PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(RATES)\") failed, error code=%d at
    DatabaseClose(db);
    return;
   }
//--- RATES 테이블에 바를 추가하기 위해 매개변수화된 요청을 생성
  string sql="INSERT INTO RATES (SYMBOL,TIME,OPEN,HIGH,LOW,CLOSE,TICK_VOLUME,SPREAD,RE
             " VALUES (?1,?2,?3,?4,?5,?6,?7,?8,?9)"; // 매개변수 요청
  int request=DatabasePrepare(db, sql);
  if(request==INVALID_HANDLE)
   {
    PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
    Print("SQL request: ", sql);
    DatabaseClose(db);
    return;
   }
//--- 첫 번째 요청 매개변수의 값을 설정
  DatabaseBind(request, 0, _Symbol);
//--- RATES 테이블에 막대를 추가하기 전에 시작 시간을 기억하세요
  start=GetMicrosecondCount();
  DatabaseTransactionBegin(db);
  int total=ArraySize(rates);
  bool request_error=false;
  for(int i=0; i<total; i++)
   {
 //--- 항목을 추가하기 전에 나머지 매개변수의 값을 설정
    ResetLastError();
    if(!DatabaseBind(request, 1, rates[i].time))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
 //--- 이전 DatabaseBind() 호출이 성공한 경우 다음 매개변수를 설정
    if(!request_error && !DatabaseBind(request, 2, rates[i].open))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 3, rates[i].high))
     {
      PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
      PrintFormat("Bar #%d line=%d", i+1, __LINE__);
      request_error=true;
      break;
     }
    if(!request_error && !DatabaseBind(request, 4, rates[i].low))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Bar #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 5, rates[i].close))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Bar #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 6, rates[i].tick_volume))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Bar #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 7, rates[i].spread))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Bar #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 8, rates[i].real_volume))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Bar #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }

  //--- 항목 삽입 요청을 실행하고 오류를 확인
    if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_MO
      {
       PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
  //--- 다음 매개변수 업데이트 전에 요청 재설정
    if(!request_error && !DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
```

© 2000-2025, MetaQuotes Ltd.

```
      break;
     }
 } //--- 모든 바를 통과

//--- 트랜잭션 상태
  if(request_error)
   {
    PrintFormat("Table RATES: failed to add %d bars ", ArraySize(rates));
    DatabaseTransactionRollback(db);
    DatabaseClose(db);
    return;
   }
  else
   {
    DatabaseTransactionCommit(db);
    PrintFormat("Table RATES: added %d bars in %d ms",
                ArraySize(rates), (GetMicrosecondCount()-start)/1000);
   }
//--- RATES 테이블을 CSV 파일로 저장
  string csv_filename=Symbol()+".csv";
  long saved=DatabaseExport(db, "SELECT * FROM RATES", csv_filename, DATABASE_EXPORT_
  if(saved>0)
    Print("Table RATES saved in ", Symbol(), ".csv");
  else
    Print("DatabaseExport() failed. Error ", GetLastError());
//--- 데이터 베이스 파일을 닫고 알리기
  DatabaseClose(db);
  PrintFormat("Database: %s created and closed", filename);
```

추가 참조

DatabasePrint, DatabaseImport

© 2000-2025, MetaQuotes Ltd.

DatabasePrint

엑스퍼트 저널에 테이블 또는 SQL 요청 실행 결과를 인쇄.

```
long  DatabasePrint(
   int           database,          // DatabaseOpen에서 수신된 데이터베이스 핸들
   const string  table_or_sql,      // 테이블 또는 SQL 요청
   uint          flags              // 플래그의 조합
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

table_or_sql

[in] 엑스퍼트 저널에 결과가 표시되는 SQL 요청 텍스트 또는 테이블 이름.

flags

[in] 출력 형식을 정의하는 플래그 조합. 플래그는 다음과 같이 정의됩니다:

DATABASE_PRINT_NO_HEADER - 테이블 열 이름(필드이름) 표시 안함
DATABASE_PRINT_NO_INDEX - 문자열 인덱스 표시 안 함
DATABASE_PRINT_NO_FRAME - 헤더와 데이터를 구분하여 프레임을 표시하지 않음
DATABASE_PRINT_STRINGS_RIGHT - 문자열을 우측으로 정렬

flags=0이면 열과 문자열이 표시되고, 헤더와 데이터는 프레임으로 구분되며, 문자열은 좌측에 정렬됩니다.

반환 값

내보낸 문자열 수를 반환하거나 오류 발생 시 -1을 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INTERNAL_ERROR (4001) - 심각한 런타임 오류;
- ERR_NOT_ENOUGH_MEMORY (4004) - 충분하지 않은 메모리;
- ERR_DATABASE_INTERNAL (5120) - 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;

참고

저널 요청 결과가 표시되면 SQL 요청은 "SELECT" 또는 "select"로 시작해야 합니다. 즉, SQL 요청에서 데이터베이스 상태를 변경할 수 없으며, 그렇지 않으면 DatabasePrint()가 실패하고 오류가 발생합니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   string filename="departments.sqlite";
//--- 공통 터미널 폴더에서 데이터베이스 생성 또는 열기
```

© 2000-2025, MetaQuotes Ltd.

```
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATAB
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드와 함께 열기 실패 ", GetLastError());
      return;
     }

//--- COMPANY 테이블을 생성
   if(!CreataTableCompany(db))
     {
      DatabaseClose(db);
      return;
     }
//--- DEPARTMENT 테이블을 생성
   if(!CreataTableDepartment(db))
     {
      DatabaseClose(db);
      return;
     }

//--- COMPANY 및 DEPARTMENT 테이블에 모든 필드 목록을 표시
   PrintFormat("요청 인쇄 시도 ₩PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMEN
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)", 0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO()\") failed, error code=%d", GetL
      DatabaseClose(db);
      return;
     }
//--- 로그에 COMPANY 테이블을 표시
   PrintFormat("요청 인쇄 시도 ₩SELECT * from COMPANY₩");
   if(DatabasePrint(db, "SELECT * from COMPANY", 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- COMPANY 및 DEPARTMENT 테이블을 결합하기 위한 요청 텍스트
   string request="SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT "
                  "ON COMPANY.ID = DEPARTMENT.EMP_ID";
//--- 테이블 조합 결과를 표시
   PrintFormat("요청 인쇄 시도 ₩SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN
   if(DatabasePrint(db, request, 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- 데이터베이스 닫기
   DatabaseClose(db);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
/*
결과:
Try to print request "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)"
#| cid name    type     notnull dflt_value pk
-+----------------------------------------
1|   0 ID      INT            1             1
2|   1 NAME    TEXT           1             0
3|   2 AGE     INT            1             0
4|   3 ADDRESS CHAR(50)       0             0
5|   4 SALARY  REAL           0             0
#| cid name   type     notnull dflt_value pk
-+---------------------------------------
1|   0 ID     INT            1             1
2|   1 DEPT   CHAR(50)       1             0
3|   2 EMP_ID INT            1             0
Try to print request "SELECT * from COMPANY"
#| ID NAME  AGE ADDRESS     SALARY
-+-------------------------------
1|  1 Paul   32 California 25000.0
2|  2 Allen  25 Texas      15000.0
3|  3 Teddy  23 Norway     20000.0
4|  4 Mark   25 Rich-Mond  65000.0
5|  5 David  27 Texas      85000.0
6|  6 Kim    22 South-Hall 45000.0
7|  7 James  24 Houston    10000.0
Try to print request "SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMEN
#| EMP_ID NAME  DEPT
-+-------------------------
1|      1 Paul  IT Billing
2|      2 Allen Engineering
3|        Teddy
4|        Mark
5|        David
6|        Kim
7|      7 James Finance
*/
//+------------------------------------------------------------------+
//| COMPANY 테이블 생성                                              |
//+------------------------------------------------------------------+
bool CreateTableCompany(int database)
  {
//--- COMPANY 테이블이 있는 경우 삭제
   if(DatabaseTableExists(database, "COMPANY"))
     {
      //--- 테이블 삭제
      if(!DatabaseExecute(database, "DROP TABLE COMPANY"))
        {
         Print("코드가 있는 테이블 COMPANY 삭제 실패 ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
         return(false);
        }
      }
//--- COMPANY 테이블을 생성
   if(!DatabaseExecute(database, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
      {
       Print("DB: 코드가 있는 테이블 COMPANY 삭제 실패 ", GetLastError());
       return(false);
      }

//--- COMPANY 테이블에 데이터 입력
   if(!DatabaseExecute(database, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VAI
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (5, 'I
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'P
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'J
      {
       Print("코드로 COMPANY 삽입 실패 ", GetLastError());
       return(false);
      }
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| DEPARTMENT 테이블 생성                                           |
//+------------------------------------------------------------------+
bool CreateTableDepartment(int database)
  {
//--- DEPARTMENT 테이블이 있는 경우 삭제
   if(DatabaseTableExists(database, "DEPARTMENT"))
     {
      //--- 테이블 삭제
      if(!DatabaseExecute(database, "DROP TABLE DEPARTMENT"))
        {
         Print("코드로 테이블 DEPARTMENT를 삭제 실패 ", GetLastError());
         return(false);
        }
      }
//--- DEPARTMENT 테이블을 생성
   if(!DatabaseExecute(database, "CREATE TABLE DEPARTMENT ("
                       "ID      INT PRIMARY KEY   NOT NULL,"
                       "DEPT    CHAR(50)          NOT NULL,"
```

© 2000-2025, MetaQuotes Ltd.

```
                        "EMP_ID  INT                NOT NULL);"))
     {
      Print("DB: 코드로 테이블 DEPARTMENT 생성 실패 ", GetLastError());
      return(false);
     }

//--- DEPARTMENT 테이블에 데이터를 입력
   if(!DatabaseExecute(database, "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (1, '
                       "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (2, 'Engineerin
                       "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (3, 'Finance',
     {
      Print("코드로 DEPARTMENT 삽입 실패 ", GetLastError());
      return(false);
     }
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------
```

추가 참조

DatabaseExport, DatabaseImport

© 2000-2025, MetaQuotes Ltd.

DatabaseTableExists

데이터베이스에 테이블이 있는지 확인.

```
bool  DatabaseTableExists(
   int     database,      // DatabaseOpen에서 수신된 데이터베이스 핸들
   string  table          // 테이블 이름
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

table

[in] 테이블 이름.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INVALID_PARAMETER (4003) - 테이블 이름이 지정되지 않음(빈 문자열 또는 NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - 요청을 UTF-8 문자열로 변환 중에 오류 발생;
- ERR_DATABASE_INTERNAL (5120) - 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_EXECUTE (5124) - 요청 실행 오류;
- ERR_DATABASE_NO_MORE_DATA (5126) - 테이블이 존재하지 않음 (오류가 아니라, 정상적 완료).

추가 참조

DatabasePrepare, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseExecute

지정된 데이터베이스에 대한 요청을 실행.

```
bool  DatabaseExecute(
   int      database,       // DatabaseOpen에서 수신된 데이터베이스 핸들
   string   sql             // SQL 요청
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

sql

[in] SQL 요청.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INTERNAL_ERROR (4001) - 심각한 런타임 오류;
- ERR_INVALID_PARAMETER (4003) - sql 매개변수의 문자열이 비어있음;
- ERR_NOT_ENOUGH_MEMORY (4004) - 충분하지 않은 메모리;
- ERR_WRONG_STRING_PARAMETER (5040) - 요청을 UTF-8 문자열로 변환 중에 오류 발생;
- ERR_DATABASE_INTERNAL (5120) - 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_EXECUTE (5124) - 요청 실행 오류.

예:

```
//--- 심볼 통계
struct Symbol_Stats
  {
   string            name;             // 심볼 이름
   int               trades;           // 심볼에 대한 거래 수
   double            gross_profit;     // 심볼에 대한 총 이익
   double            gross_loss;       // 심볼에 대한 총 손실
   double            total_commission; // 심볼에 대한 총 수수료
   double            total_swap;       // 심볼에 대한 총 스왑
   double            total_profit;     // 스왑 및 커미션을 제외한 총액
   double            net_profit;       // 변동 및 수수료를 고려한 순이익
   int               win_trades;       // 수익 거래의 수
   int               loss_trades;      // 손실 거래의 수
   double            expected_payoff;  // 스왑 및 수수료를 제외한 거래의 예상 수익
   double            win_percent;      // 매매의 승률
   double            loss_percent;     // 손실 거래의 %
   double            average_profit;   // 평균 수익
   double            average_loss;     // 평균 손실
   double            profit_factor;    // 수익 계수
```

© 2000-2025, MetaQuotes Ltd.

```
  };

//--- 매직 넘버 통계
struct Magic_Stats
  {
   long              magic;            // EA의 매직 넘버
   int               trades;           // 심볼에 대한 거래 수
   double            gross_profit;     // 심볼에 대한 총 이익
   double            gross_loss;       // 심볼에 대한 총 손실
   double            total_commission; // 심볼에 대한 총 수수료
   double            total_swap;       // 심볼에 대한 총 스왑
   double            total_profit;     // 스왑 및 커미션을 제외한 총액
   double            net_profit;       // 변동 및 수수료를 고려한 순이익
   int               win_trades;       // 수익 거래의 수
   int               loss_trades;      // 손실 거래의 수
   double            expected_payoff;  // 스왑 및 수수료를 제외한 거래의 예상 수익
   double            win_percent;      // 매매의 승률
   double            loss_percent;     // 손실 거래의 %
   double            average_profit;   // 평균 수익
   double            average_loss;     // 평균 손실
   double            profit_factor;    // 수익 계수
  };

//--- 진입 시간 통계
struct Hour_Stats
  {
   char              hour_in;          // 시장 진입 시간
   int               trades;           // 이 진입 시간의 거래 수
   double            volume;           // 이 진입 시간의 거래 볼륨
   double            gross_profit;     // 이 진입 시간의 총 이익
   double            gross_loss;       // 이 진입 시간의 총 손실
   double            net_profit;       // 변동 및 수수료를 고려한 순이익
   int               win_trades;       // 수익 거래의 수
   int               loss_trades;      // 손실 거래의 수
   double            expected_payoff;  // 스왑 및 수수료를 제외한 거래의 예상 수익
   double            win_percent;      // 매매의 승률
   double            loss_percent;     // 손실 거래의 %
   double            average_profit;   // 평균 수익
   double            average_loss;     // 평균 손실
   double            profit_factor;    // 수익 계수
  };

int ExtDealsTotal=0;;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 파일 이름 생성
```

© 2000-2025, MetaQuotes Ltd.

```
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_stats.sqlite";
//--- 공용 터미널 폴더에서 데이터베이스 열기/생성
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드와 함께 열기 실패 ", GetLastError());
      return;
     }
//--- DEALS 표를 작성
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
   PrintFormat("거래 내역: %d ", ExtDealsTotal);

//--- 심볼당 트레이딩 통계 가져오기
   int request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then (r.gross_profit+r.gro
                               "   (case when r.trades != 0 then r.win_trades*100.0/r.
                               "   (case when r.trades != 0 then r.loss_trades*100.0/r
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-
                               "FROM "
                               "   ("
                               "   SELECT SYMBOL,"
                               "   sum(case when entry =1 then 1 else 0 end) as trades
                               "   sum(case when profit > 0 then profit else 0 end) as
                               "   sum(case when profit < 0 then profit else 0 end) as
                               "   sum(swap) as total_swap,"
                               "   sum(commission) as total_commission,"
                               "   sum(profit) as total_profit,"
                               "   sum(profit+swap+commission) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as win_
                               "   sum(case when profit < 0 then 1 else 0 end) as loss
                               "   FROM DEALS "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY SYMBOL"
                               "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드를 이용한 요청 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Symbol_Stats stats[], symbol_stats;
   ArrayResize(stats, ExtDealsTotal);
   int i=0;
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 요청 결과에서 레코드 가져오기
   for(; DatabaseReadBind(request, symbol_stats) ; i++)
     {
      stats[i].name=symbol_stats.name;
      stats[i].trades=symbol_stats.trades;
      stats[i].gross_profit=symbol_stats.gross_profit;
      stats[i].gross_loss=symbol_stats.gross_loss;
      stats[i].total_commission=symbol_stats.total_commission;
      stats[i].total_swap=symbol_stats.total_swap;
      stats[i].total_profit=symbol_stats.total_profit;
      stats[i].net_profit=symbol_stats.net_profit;
      stats[i].win_trades=symbol_stats.win_trades;
      stats[i].loss_trades=symbol_stats.loss_trades;
      stats[i].expected_payoff=symbol_stats.expected_payoff;
      stats[i].win_percent=symbol_stats.win_percent;
      stats[i].loss_percent=symbol_stats.loss_percent;
      stats[i].average_profit=symbol_stats.average_profit;
      stats[i].average_loss=symbol_stats.average_loss;
      stats[i].profit_factor=symbol_stats.profit_factor;
     }
   ArrayResize(stats, i);
   Print("심볼별 거래 통계");
   ArrayPrint(stats);
   Print("");
//--- 요청 삭제
   DatabaseFinalize(request);

//--- 매직 넘버별 엑스퍼트 어드바이저의 트레이딩 통계 가져오기
   request=DatabasePrepare(db, "SELECT r.*,"
                           "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                           "   (case when r.trades != 0 then r.win_trades*100.0/r.trad
                           "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                           "   r.gross_profit/r.win_trades as average_profit,"
                           "   r.gross_loss/r.loss_trades as average_loss,"
                           "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                           "FROM "
                           "   ("
                           "   SELECT MAGIC,"
                           "   sum(case when entry =1 then 1 else 0 end) as trades,"
                           "   sum(case when profit > 0 then profit else 0 end) as gro
                           "   sum(case when profit < 0 then profit else 0 end) as gro
                           "   sum(swap) as total_swap,"
                           "   sum(commission) as total_commission,"
                           "   sum(profit) as total_profit,"
                           "   sum(profit+swap+commission) as net_profit,"
                           "   sum(case when profit > 0 then 1 else 0 end) as win_trad
                           "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                           "   FROM DEALS "
                           "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
```

© 2000-2025, MetaQuotes Ltd.

```
                              "   GROUP BY MAGIC"
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드를 이용한 요청 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Magic_Stats EA_stats[], magic_stats;
   ArrayResize(EA_stats, ExtDealsTotal);
   i=0;
//--- 인쇄
   for(; DatabaseReadBind(request, magic_stats) ; i++)
     {
      EA_stats[i].magic=magic_stats.magic;
      EA_stats[i].trades=magic_stats.trades;
      EA_stats[i].gross_profit=magic_stats.gross_profit;
      EA_stats[i].gross_loss=magic_stats.gross_loss;
      EA_stats[i].total_commission=magic_stats.total_commission;
      EA_stats[i].total_swap=magic_stats.total_swap;
      EA_stats[i].total_profit=magic_stats.total_profit;
      EA_stats[i].net_profit=magic_stats.net_profit;
      EA_stats[i].win_trades=magic_stats.win_trades;
      EA_stats[i].loss_trades=magic_stats.loss_trades;
      EA_stats[i].expected_payoff=magic_stats.expected_payoff;
      EA_stats[i].win_percent=magic_stats.win_percent;
      EA_stats[i].loss_percent=magic_stats.loss_percent;
      EA_stats[i].average_profit=magic_stats.average_profit;
      EA_stats[i].average_loss=magic_stats.average_loss;
      EA_stats[i].profit_factor=magic_stats.profit_factor;
     }
   ArrayResize(EA_stats, i);
   Print("매직 넘버별 거래 통계");
   ArrayPrint(EA_stats);
   Print("");
//--- 요청 삭제
   DatabaseFinalize(request);

//--- 오픈 포지션 관리를 위한 헤징 시스템을 계좌에서 사용하는지 확인
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- 거래를 트랜잭션을 통한 단순한 방법을 사용하여 트레이딩으로 전환할 수 없으므로 완전한
      DatabaseClose(db);
      return;
     }

//--- 이제 DEALS 표를 기반으로 TRADES 표를 작성
   if(!CreateTableTrades(db))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      DatabaseClose(db);
      return;
     }
//--- DEALS 표 데이터를 기반으로 SQL 쿼리를 사용하여 TRADES 표를 채우기
   if(DatabaseTableExists(db, "DEALS"))
     //--- TRADES 표 덧붙이기
     if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,HOUR_IN,TICKET,TYPE,VOLUME,S
                             "SELECT "
                             "   d1.time as time_in,"
                             "   d1.hour as hour_in,"
                             "   d1.position_id as ticket,"
                             "   d1.type as type,"
                             "   d1.volume as volume,"
                             "   d1.symbol as symbol,"
                             "   d1.price as price_in,"
                             "   d2.time as time_out,"
                             "   d2.price as price_out,"
                             "   d1.commission+d2.commission as commission,"
                             "   d2.swap as swap,"
                             "   d2.profit as profit "
                             "FROM DEALS d1 "
                             "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                             "WHERE d1.entry=0 AND d2.entry=1      "))
        {
         Print("DB: TRADES 테이블을 코드로 채우는데 실패 ", GetLastError());
         return;
        }

//--- 시장 진입 시간별 거래 통계 가져오기
   request=DatabasePrepare(db, "SELECT r.*,"
                               "   (case when r.trades != 0 then (r.gross_profit+r.gross_l
                               "   (case when r.trades != 0 then r.win_trades*100.0/r.trad
                               "   (case when r.trades != 0 then r.loss_trades*100.0/r.tra
                               "   r.gross_profit/r.win_trades as average_profit,"
                               "   r.gross_loss/r.loss_trades as average_loss,"
                               "   (case when r.gross_loss!=0.0 then r.gross_profit/(-r.gr
                               "FROM "
                               "   ("
                               "   SELECT HOUR_IN,"
                               "   count() as trades,"
                               "   sum(volume) as volume,"
                               "   sum(case when profit > 0 then profit else 0 end) as gro
                               "   sum(case when profit < 0 then profit else 0 end) as gro
                               "   sum(profit) as net_profit,"
                               "   sum(case when profit > 0 then 1 else 0 end) as win_trad
                               "   sum(case when profit < 0 then 1 else 0 end) as loss_tra
                               "   FROM TRADES "
                               "   WHERE SYMBOL <> '' and SYMBOL is not NULL "
                               "   GROUP BY HOUR_IN"
```

© 2000-2025, MetaQuotes Ltd.

```
                              "   ) as r");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드를 이용한 요청 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }
   Hour_Stats hours_stats[], h_stats;
   ArrayResize(hours_stats, ExtDealsTotal);
   i=0;
//--- 인쇄
   for(; DatabaseReadBind(request, h_stats) ; i++)
     {
      hours_stats[i].hour_in=h_stats.hour_in;
      hours_stats[i].trades=h_stats.trades;
      hours_stats[i].volume=h_stats.volume;
      hours_stats[i].gross_profit=h_stats.gross_profit;
      hours_stats[i].gross_loss=h_stats.gross_loss;
      hours_stats[i].net_profit=h_stats.net_profit;
      hours_stats[i].win_trades=h_stats.win_trades;
      hours_stats[i].loss_trades=h_stats.loss_trades;
      hours_stats[i].expected_payoff=h_stats.expected_payoff;
      hours_stats[i].win_percent=h_stats.win_percent;
      hours_stats[i].loss_percent=h_stats.loss_percent;
      hours_stats[i].average_profit=h_stats.average_profit;
      hours_stats[i].average_loss=h_stats.average_loss;
      hours_stats[i].profit_factor=h_stats.profit_factor;
     }
   ArrayResize(hours_stats, i);
   Print("Trade statistics by entry hour");
   ArrayPrint(hours_stats);
   Print("");
//--- 요청 삭제
   DatabaseFinalize(request);

//--- 데이터베이스 닫기
   DatabaseClose(db);
   return;
  }
/*
Deals in the trading history: 2771
Trade statistics by Symbol
      [name] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tot
[0] "AUDUSD"      112      503.20000   -568.00000           -8.83000    -24.64000
[1] "EURCHF"      125      607.71000   -956.85000          -11.77000    -45.02000
[2] "EURJPY"      127     1078.49000  -1057.83000          -10.61000    -45.76000
[3] "EURUSD"      233     1685.60000  -1386.80000          -41.00000    -83.76000
[4] "GBPCHF"      125     1881.37000  -1424.72000          -22.60000    -51.56000
[5] "GBPJPY"      127     1943.43000  -1776.67000          -18.84000    -52.46000
```

© 2000-2025, MetaQuotes Ltd.

```
[6] "GBPUSD"      121     1668.50000  -1438.20000           -7.96000    -49.93000
[7] "USDCAD"       99      405.28000   -475.47000           -8.68000    -31.68000
[8] "USDCHF"      206     1588.32000  -1241.83000          -17.98000    -65.92000
[9] "USDJPY"      107      464.73000   -730.64000          -35.12000    -34.24000

Trade statistics by Magic Number
    [magic] [trades] [gross_profit] [gross_loss] [total_commission] [total_swap] [tota
[0]     100      242     2584.80000  -2110.00000          -33.36000    -93.53000
[1]     200      254     3021.92000  -2834.50000          -29.45000    -98.22000
[2]     300      250     2489.08000  -2381.57000          -34.37000    -96.58000
[3]     400      224     1272.50000  -1283.00000          -24.43000    -64.80000
[4]     500      198     1141.23000  -1051.91000          -27.66000    -63.36000
[5]     600      214     1317.10000  -1396.03000          -34.12000    -68.48000

Trade statistics by entry hour
     [hour_in] [trades] [volume] [gross_profit] [gross_loss] [net_profit] [win_trades]
[ 0]         0       50  5.00000      336.51000   -747.47000   -410.96000           21
[ 1]         1       20  2.00000      102.56000    -57.20000     45.36000           12
[ 2]         2        6  0.60000       38.55000    -14.60000     23.95000            5
[ 3]         3       38  3.80000      173.84000   -200.15000    -26.31000           22
[ 4]         4       60  6.00000      361.44000   -389.40000    -27.96000           27
[ 5]         5       32  3.20000      157.43000   -179.89000    -22.46000           20
[ 6]         6       18  1.80000       95.59000   -162.33000    -66.74000           11
[ 7]         7       14  1.40000       38.48000   -134.30000    -95.82000            9
[ 8]         8       42  4.20000      368.48000   -322.30000     46.18000           24
[ 9]         9      118 11.80000     1121.62000   -875.21000    246.41000           72
[10]        10      206 20.60000     2280.59000  -2021.80000    258.79000          115
[11]        11      138 13.80000     1377.02000   -994.18000    382.84000           84
[12]        12      152 15.20000     1247.56000  -1463.80000   -216.24000           84
[13]        13       64  6.40000      778.27000   -516.22000    262.05000           36
[14]        14       62  6.20000      536.93000   -427.47000    109.46000           38
[15]        15       50  5.00000      699.92000   -413.00000    286.92000           28
[16]        16       88  8.80000      778.55000   -514.00000    264.55000           51
[17]        17       76  7.60000      533.92000  -1019.46000   -485.54000           44
[18]        18       52  5.20000      237.17000   -246.78000     -9.61000           24
[19]        19       52  5.20000      407.67000   -150.36000    257.31000           30
[20]        20       18  1.80000       65.92000    -89.09000    -23.17000            9
[21]        21       10  1.00000       41.86000    -32.38000      9.48000            7
[22]        22       14  1.40000       45.55000    -83.72000    -38.17000            6
[23]        23        2  0.20000        1.20000     -1.90000     -0.70000            1
*/

//+------------------------------------------------------------------+
//| DEALS 표 생성                                                    |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- DEALS 표가 이미 있는 경우 삭제
   if(!DeleteTable(database, "DEALS"))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       return(false);
      }
//--- 표의 존재유무 확인
   if(!DatabaseTableExists(database, "DEALS"))
      //--- 표 생성
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "HOUR        INT,"
                          "REASON      INT);"))
        {
         Print("DB: 코드로 DEALS 표를 생성하는데 실패 ", GetLastError());
         return(false);
        }
//---  전체 트레이딩 내역을 요청
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- 거래 내역을 지정 간격 내에 요청
   HistorySelect(from_date, to_date);
   ExtDealsTotal=HistoryDealsTotal();
//--- 표에 거래 추가
   if(!InsertDeals(database))
      return(false);
//--- 표가 성공적으로 생성되었습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 데이터베이스에서 지정된 이름의 표를 삭제                         |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("Failed to drop the DEALS table with code ", GetLastError());
      return(false);
     }
//--- 테이블이 성공적으로 삭제되었습니다
```

© 2000-2025, MetaQuotes Ltd.

```
   return(true);
  }
//+------------------------------------------------------------------+
//| 데이터베이스 표에 거래 추가                                      |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- 보조 변수
   ulong    deal_ticket;          // 거래 티켓
   long     order_ticket;         // 거래가 실행된 주문 티켓
   long     position_ticket;      // 거래가 속한 포지션의 ID
   datetime time;                 // 거래 실행 시간
   long     type ;                // 거래 유형
   long     entry ;               // 거래 방향
   string   symbol;               // 거래가 실행된 심볼
   double   volume;               // 작업 볼륨
   double   price;                // 가격
   double   profit;               // 재정적 결과
   double   swap;                 // 스왑
   double   commission;           // 수수료
   long     magic;                // 매직 넘버 (엑스퍼트 어드바이저 ID)
   long     reason;               // 거래 실행 이유 또는 소스
   char     hour;                 // 거래 실행 시간
   MqlDateTime time_strusture;
//--- 모든 거래를 검토하여 데이터 베이스에 추가
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- 트랜잭션을 실행하기 전에 데이터베이스를 잠금
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      TimeToStruct(time, time_strusture);
      hour= (char)time_strusture.hour;
      //--- 다음 요청에 따라 각 거래를 표에 추가
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIN
```

© 2000-2025, MetaQuotes Ltd.

```
                                        "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                        deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: %d 코드로 #%d 거래 삽입하는데 실패", __FUNCTION__, deal_ticket,
         PrintFormat("i=%d: 거래 #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- 트랜잭션 실행 오류를 확인
   if(failed)
     {
      //--- 모든 트랜잭션 롤백 및 데이터베이스 잠금 해제
      DatabaseTransactionRollback(database);
      PrintFormat("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastErro
      return(false);
     }
//--- 모든 트랜잭션이 성공적으로 수행됨 - 변경 사항을 기록하고 데이터베이스 잠금을 해제합니다
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| TRADES 표를 생성                                                 |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- TRADES 표가 이미 있는 경우 삭제
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- 표의 존재유무 확인
   if(!DatabaseTableExists(database, "TRADES"))
      //--- 표 생성
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "HOUR_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: 코드로 TRADES 표를 생성하는데 실패 ", GetLastError());
         return(false);
```

© 2000-2025, MetaQuotes Ltd.

```
         }
//--- 표가 성공적으로 생성되었습니다
   return(true);
  }
//+------------------------------------------------------------------+
```

추가 참조

DatabasePrepare, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabasePrepare

실행할 수 있는 요청 핸들을 DatabaseRead()를 사용하여 생성합니다.

```
int  DatabasePrepare(
   int      database,       // DatabaseOpen에서 수신된 데이터베이스 핸들
   string   sql,            // SQL 요청
            ...             // 매개변수 요청
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

sql

[in] 이름이 지정된 자동 대체 매개변수를 포함할 수 있는 SQL 요청 ?1,?2,...

...

[in] 자동으로 대체되는 요청 매개변수.

반환 값

성공하면 함수는 SQL 요청에 대한 핸들을 반환합니다. 그렇지 않으면 INVALID_HANDLE을 반환합니다. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INVALID_PARAMETER (4003) - 데이터베이스 파일의 경로에 빈 문자열이 포함되어 있거나 호환되지 않는 플래그 조합이 설정;
- ERR_NOT_ENOUGH_MEMORY (4004) - 충분하지 않은 메모리;
- ERR_WRONG_STRING_PARAMETER (5040) - 요청을 UTF-8 문자열로 변환 중에 오류 발생;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - 허용되는 최대 데이터베이스 개체 수를 초과;
- ERR_DATABASE_PREPARE (5125) - 생성 오류 요청.

참고

DatabasePrepare() 기능은 데이터베이스에 대한 요청을 수행하지 않습니다. 이 목적은 요청 매개변수를 확인하고 확인 결과에 따라 SQL 요청을 실행하기 위한 핸들을 반환하는 것입니다. 요청 자체는 DatabaseRead()를 처음 호출할 때 설정됩니다.

예:

```
//--- 거래를 저장할 구조
struct Deal
  {
   ulong             ticket;           // DEAL_TICKET
   long              order_ticket;     // DEAL_ORDER
   long              position_ticket;  // DEAL_POSITION_ID
   datetime          time;             // DEAL_TIME
   char              type;             // DEAL_TYPE
   char              entry;            // DEAL_ENTRY
   string            symbol;           // DEAL_SYMBOL
   double            volume;           // DEAL_VOLUME
```

© 2000-2025, MetaQuotes Ltd.

```
   double            price;            // DEAL_PRICE
   double            profit;           // DEAL_PROFIT
   double            swap;             // DEAL_SWAP
   double            commission;       // DEAL_COMMISSION
   long              magic;            // DEAL_MAGIC
   char              reason;           // DEAL_REASON
  };
//--- 거래 저장 구조: 회원의 주문은 터미널 내 포지션에 해당
struct Trade
  {
   datetime          time_in;          // 진입 시간
   ulong             ticket;           // 포지션 ID
   char              type;             // 구매 또는 판매
   double            volume;           // 볼륨
   string            symbol;           // 심볼
   double            price_in;         // 진입 가격
   datetime          time_out;         // 퇴장 시간
   double            price_out;        // 퇴장 가격
   double            commission;       // 진입 및 퇴장 수수료
   double            swap;             // 스왑
   double            profit;           // 수익 또는 손실
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 파일 이름 생성
   string filename=IntegerToString(AccountInfoInteger(ACCOUNT_LOGIN))+"_trades.sqlite"
//--- 공용 터미널 폴더에서 데이터베이스 열기/생성
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드와 함께 열기 실패 ", GetLastError());
      return;
     }
//--- DEALS 표를 작성
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  전체 트레이딩 내역을 요청
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- 거래 내역을 지정 간격 내에 요청
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("트레이딩 내역 내 거래: %d ", deals_total);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 표에 거래 추가
   if(!InsertDeals(db))
      return;
//--- 첫 10개 거래 표시
   Deal deals[], deal;
   ArrayResize(deals, 10);
   int request=DatabasePrepare(db, "SELECT * FROM DEALS");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드를 이용한 요청 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }
   int i;
   for(i=0; DatabaseReadBind(request, deal); i++)
     {
      if(i>=10)
         break;
      deals[i].ticket=deal.ticket;
      deals[i].order_ticket=deal.order_ticket;
      deals[i].position_ticket=deal.position_ticket;
      deals[i].time=deal.time;
      deals[i].type=deal.type;
      deals[i].entry=deal.entry;
      deals[i].symbol=deal.symbol;
      deals[i].volume=deal.volume;
      deals[i].price=deal.price;
      deals[i].profit=deal.profit;
      deals[i].swap=deal.swap;
      deals[i].commission=deal.commission;
      deals[i].magic=deal.magic;
      deals[i].reason=deal.reason;
     }
//--- 거래 출력
   if(i>0)
     {
      ArrayResize(deals, i);
      PrintFormat("첫 %d 거래:", i);
      ArrayPrint(deals);
     }

//--- 사용 후 요청 삭제
   DatabaseFinalize(request);

//--- 오픈 포지션 관리를 위한 헤징 시스템을 계좌에서 사용하는지 확인
   if((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)!=ACCOUNT_MARGI
     {
      //--- 거래를 트랜잭션을 통한 단순한 방법을 사용하여 트레이딩으로 전환할 수 없으므로 완전한
      DatabaseClose(db);
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }

//--- 이제 DEALS 표를 기반으로 TRADES 표를 작성
   if(!CreateTableTrades(db))
     {
      DatabaseClose(db);
      return;
     }
//--- DEALS 표 데이터를 기반으로 SQL 쿼리를 사용하여 TRADES 표를 채우기
   ulong start=GetMicrosecondCount();
   if(DatabaseTableExists(db, "DEALS"))
      //--- TRADES 표 덧붙이기
      if(!DatabaseExecute(db, "INSERT INTO TRADES(TIME_IN,TICKET,TYPE,VOLUME,SYMBOL,PR
                          "SELECT "
                          "   d1.time as time_in,"
                          "   d1.position_id as ticket,"
                          "   d1.type as type,"
                          "   d1.volume as volume,"
                          "   d1.symbol as symbol,"
                          "   d1.price as price_in,"
                          "   d2.time as time_out,"
                          "   d2.price as price_out,"
                          "   d1.commission+d2.commission as commission,"
                          "   d2.swap as swap,"
                          "   d2.profit as profit "
                          "FROM DEALS d1 "
                          "INNER JOIN DEALS d2 ON d1.position_id=d2.position_id "
                          "WHERE d1.entry=0 AND d2.entry=1     "))
        {
         Print("DB: 코드로 TRADES 표를 채우는데 실패 ", GetLastError());
         return;
        }
   ulong transaction_time=GetMicrosecondCount()-start;

//--- 첫 10개 거래 표시
   Trade trades[], trade;
   ArrayResize(trades, 10);
   request=DatabasePrepare(db, "SELECT * FROM TRADES");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드를 이용한 요청 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }
   for(i=0; DatabaseReadBind(request, trade); i++)
     {
      if(i>=10)
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
      trades[i].time_in=trade.time_in;
      trades[i].ticket=trade.ticket;
      trades[i].type=trade.type;
      trades[i].volume=trade.volume;
      trades[i].symbol=trade.symbol;
      trades[i].price_in=trade.price_in;
      trades[i].time_out=trade.time_out;
      trades[i].price_out=trade.price_out;
      trades[i].commission=trade.commission;
      trades[i].swap=trade.swap;
      trades[i].profit=trade.profit;
     }
//--- 거래 인쇄
   if(i>0)
     {
      ArrayResize(trades, i);
      PrintFormat("\r\nThe first %d trades:", i);
      ArrayPrint(trades);
      PrintFormat("TRADES 표를 채우는 데 %.2f 밀리초가 걸렸습니다",double(transaction_time
     }
//--- 사용 후 요청 삭제
   DatabaseFinalize(request);

//--- 데이터베이스 닫기
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2741
   The first 10 deals:
       [ticket] [order_ticket] [position_ticket]              [time] [type] [entry] [s
   [0] 34429573              0                 0 2019.09.05 22:39:59      2        0 ""
   [1] 34432127       51447238          51447238 2019.09.06 06:00:03      0        0 "U
   [2] 34432128       51447239          51447239 2019.09.06 06:00:03      1        0 "U
   [3] 34432450       51447565          51447565 2019.09.06 07:00:00      0        0 "E
   [4] 34432456       51447571          51447571 2019.09.06 07:00:00      1        0 "A
   [5] 34432879       51448053          51448053 2019.09.06 08:00:00      1        0 "U
   [6] 34432888       51448064          51448064 2019.09.06 08:00:00      0        0 "U
   [7] 34435147       51450470          51450470 2019.09.06 10:30:00      1        0 "E
   [8] 34435152       51450476          51450476 2019.09.06 10:30:00      0        0 "C
   [9] 34435154       51450479          51450479 2019.09.06 10:30:00      1        0 "E

   The first 10 trades:
                [time_in] [ticket] [type] [volume] [symbol] [price_in]           [time
   [0] 2019.09.06 06:00:03 51447238      0  0.10000 "USDCAD"    1.32320 2019.09.06 18:
   [1] 2019.09.06 06:00:03 51447239      1  0.10000 "USDCHF"    0.98697 2019.09.06 18:
   [2] 2019.09.06 07:00:00 51447565      0  0.10000 "EURUSD"    1.10348 2019.09.09 03:
   [3] 2019.09.06 07:00:00 51447571      1  0.10000 "AUDUSD"    0.68203 2019.09.09 03:
   [4] 2019.09.06 08:00:00 51448053      1  0.10000 "USDCHF"    0.98701 2019.09.06 18:
```

© 2000-2025, MetaQuotes Ltd.

```
   [5] 2019.09.06 08:00:00 51448064      0  0.10000 "USDJPY"  106.96200 2019.09.06 18:
   [6] 2019.09.06 10:30:00 51450470      1  0.10000 "EURUSD"    1.10399 2019.09.06 14:
   [7] 2019.09.06 10:30:00 51450476      0  0.10000 "GBPUSD"    1.23038 2019.09.06 14:
   [8] 2019.09.06 10:30:00 51450479      1  0.10000 "EURJPY"  118.12000 2019.09.06 14:
   [9] 2019.09.06 10:30:00 51450480      0  0.10000 "GBPJPY"  131.65300 2019.09.06 14:
   Filling the TRADES table took 12.51 milliseconds
*/
//+------------------------------------------------------------------+
//| DEALS 표 생성                                                    |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- DEALS 표가 이미 있는 경우 삭제
   if(!DeleteTable(database, "DEALS"))
     {
      return(false);
     }
//--- 표의 존재유무 확인
   if(!DatabaseTableExists(database, "DEALS"))
      //--- 표 생성
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: 코드로 DEALS 표를 생성하는데 실패 ", GetLastError());
         return(false);
        }
//--- 표가 성공적으로 생성되었습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 데이터베이스에서 지정된 이름의 표를 삭제                         |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("Failed to drop the DEALS table  with code ", GetLastError());
      return(false);
     }
//--- 테이블이 성공적으로 삭제되었습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 데이터베이스 표에 거래 추가                                      |
//+------------------------------------------------------------------+
bool InsertDeals(int database)
  {
//--- 보조 변수
   ulong    deal_ticket;         // 거래 티켓
   long     order_ticket;        // 거래가 실행된 주문 티켓
   long     position_ticket;     // 거래가 속한 포지션의 ID
   datetime time;                // 거래 실행 시간
   long     type ;               // 거래 유형
   long     entry ;              // 거래 방향
   string   symbol;              // 거래가 실행된 심볼
   double   volume;              // 작업 볼륨
   double   price;               // 가격
   double   profit;              // 재정적 결과
   double   swap;                // 스왑
   double   commission;          // 수수료
   long     magic;               // 매직 넘버 (엑스퍼트 어드바이저 ID)
   long     reason;              // 거래 실행 이유 또는 소스
//--- 모든 거래를 검토하여 데이터 베이스에 추가
   bool failed=false;
   int deals=HistoryDealsTotal();
// --- 트랜잭션을 실행하기 전에 데이터베이스를 잠금
   DatabaseTransactionBegin(database);
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- 다음 요청에 따라 각 거래를 표에 추가
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIM
```

© 2000-2025, MetaQuotes Ltd.

```
                                        "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                        deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: %d 코드로 #%d 거래 삽입하는데 실패", __FUNCTION__, deal_ticket,
         PrintFormat("i=%d: 거래 #%d  %s", i, deal_ticket, symbol);
         failed=true;
         break;
        }
     }
//--- 트랜잭션 실행 오류를 확인
   if(failed)
     {
      //--- 모든 트랜잭션 롤백 및 데이터베이스 잠금 해제
      DatabaseTransactionRollback(database);
      PrintFormat("%s: %d 코드로 DatabaseExecute() 실패", __FUNCTION__, GetLastError())
      return(false);
     }
//--- 모든 트랜잭션이 성공적으로 수행됨 - 변경 사항을 기록하고 데이터베이스 잠금을 해제합니다
   DatabaseTransactionCommit(database);
   return(true);
  }
//+------------------------------------------------------------------+
//| TRADES 표를 생성                                                 |
//+------------------------------------------------------------------+
bool CreateTableTrades(int database)
  {
//--- TRADES 표가 이미 있는 경우 삭제
   if(!DeleteTable(database, "TRADES"))
      return(false);
//--- 표의 존재유무 확인
   if(!DatabaseTableExists(database, "TRADES"))
      //--- 표 생성
      if(!DatabaseExecute(database, "CREATE TABLE TRADES("
                          "TIME_IN     INT     NOT NULL,"
                          "TICKET      INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "VOLUME      REAL,"
                          "SYMBOL      CHAR(10),"
                          "PRICE_IN    REAL,"
                          "TIME_OUT    INT     NOT NULL,"
                          "PRICE_OUT   REAL,"
                          "COMMISSION  REAL,"
                          "SWAP        REAL,"
                          "PROFIT      REAL);"))
        {
         Print("DB: 코드로 TRADES 표를 생성하는데 실패 ", GetLastError());
         return(false);
        }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 표가 성공적으로 생성되었습니다
   return(true);
  }
//+------------------------------------------------------------------+
```

추가 참조

DatabaseExecute, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseReset

DatabasePrepare()를 호출한 후와 같이 요청을 재설정.

```
int  DatabaseReset(
   int  request      // DatabasePrepare에서 요청 핸들 수신
   );
```

매개변수

request

[in] DatabasePrepare()에서 가져온 요청 핸들.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_ERROR(5601)로 시작하는 SQLite 오류 코드.

참고

DatabaseReset() 기능은 매개변수 값이 서로 다른 요청을 여러 번 실행하기 위한 것입니다. 예를 들어 INSERT 명령을 사용하여 테이블에 대량으로 데이터를 추가하는 경우 각 항목에 대해 사용자 지정 필드 값 세트를 구성해야 합니다.

DatabasePrepare()와 달리 DatabaseReset() 호출은 SQL 명령을 사용하여 문자열을 새 요청으로 컴파일하지 않으므로 DatabaseReset()이 DatabasePrepare()보다 훨씬 빠르게 실행됩니다.

DatabaseReset()는 DatabaseBind() 기능 및/또는 DatabaseBindArray()와 함께 DatabaseRead()를 실행한 후 요청 매개변수 값을 변경하는 경우 사용됩니다. 즉, 요청 매개변수의 새 값을 설정하기 전에 (DatabaseBind/DatabaseBindArray 호출의 블록 전), DatabaseReset()을 호출하여 재설정해야 합니다. 매개변수화 된 요청 자체는 DatabasePrepare()를 사용하여 생성해야 합니다.

DatabasePrepare()와 마찬가지로, DatabaseReset()은 데이터 베이스 요청을 생성하지 않습니다. 직접 요청 실행은 DatabaseRead() 또는 DatabaseReadBind()를 호출할 때 수행됩니다.

DatabaseBind()/DatabaseBindArray()르를 호출하여 설정한 경우에는 DatabaseReset() 호출로 인해 매개변수 값이 재설정되지 않으며 즉, 매개변수가 해당 값을 유지합니다. 따라서 단일 매개변수의 값만 변경할 수 있습니다. There is no need to set all request parameters anew after calling DatabaseReset()을 호출한 후에는 모든 요청 매개변수를 새로 설정할 필요가 없습니다.

DatabaseFinalize()를 사용하여 제거된 요총의 핸들을 DatabaseReset()에 전달할 수 없습니다. 이로 인해 오류가 발행합니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 데이터베이스 열거나 생성
   string filename="symbols.sqlite";
```

© 2000-2025, MetaQuotes Ltd.

```
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드와함께 열기 실패 ", GetLastError());
      return;
     }
   else
      Print("데이터베이스: ", filename, " 성공적으로 열림");
//--- SYMBOLS 테이블이 있는 경우 삭제
   if(DatabaseTableExists(db, "SYMBOLS"))
     {
      //--- 테이블 삭제
      if(!DatabaseExecute(db, "DROP TABLE SYMBOLS"))
        {
         Print("코드가 있는 테이블 SYMBOLS을 삭제 실패 ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- SYMBOLS 테이블 생성
   if(!DatabaseExecute(db, "CREATE TABLE SYMBOLS("
                       "NAME           TEXT    NOT NULL,"
                       "DESCRIPTION    TEXT            ,"
                       "PATH           TEXT            ,"
                       "SPREAD         INT             ,"
                       "POINT          REAL    NOT NULL,"
                       "DIGITS         INT     NOT NULL,"
                       "JSON           BLOB );"))
     {
      Print("DB: ", filename, " 코트로 테이블을 생성 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- SYMBOLS 테이블에 모든 모든 필드 목록 표시
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(SYMBOLS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(SYMBOLS)\") failed, error code=%d
      DatabaseClose(db);
      return;
     }

//--- 매개변수화된 요청을 생성하여 심볼을 SYMBOLS 테이블에 추가
   string sql="INSERT INTO SYMBOLS (NAME,DESCRIPTION,PATH,SPREAD,POINT,DIGITS,JSON)"
              " VALUES (?1,?2,?3,?4,?5,?6,?7);"; // request parameters
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL 요청: ", sql);
```

© 2000-2025, MetaQuotes Ltd.

```
      DatabaseClose(db);
      return;
     }

//--- 모든 심볼을 살펴보고 SYMBOLS 테이블에 추가
   int symbols=SymbolsTotal(false);
   bool request_error=false;
   DatabaseTransactionBegin(db);
   for(int i=0; i<symbols; i++)
     {
      //--- 심볼을 추가하기 전에 매개변수의 값을 설정
      ResetLastError();
      string symbol=SymbolName(i, false);
      if(!DatabaseBind(request, 0, symbol))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      //--- 이전 DatabaseBind() 호출에 성공한 경우 다음 매개변수를 설정합니다
      if(!DatabaseBind(request, 1, SymbolInfoString(symbol, SYMBOL_DESCRIPTION)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 2, SymbolInfoString(symbol, SYMBOL_PATH)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 3, SymbolInfoInteger(symbol, SYMBOL_SPREAD)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 4, SymbolInfoDouble(symbol, SYMBOL_POINT)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 5, SymbolInfoInteger(symbol, SYMBOL_DIGITS)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
```

© 2000-2025, MetaQuotes Ltd.

```
        }
      if(!DatabaseBind(request, 6, GetSymBolAsJson(symbol)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }

      //--- 진입 입력 요청을 실행하고 오류를 확인
      if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
        {
         PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
      else
         PrintFormat("%d: added %s", i+1, symbol);
      //--- 다음 매개변수 업데이트 전에 요청을 재설정
      if(!DatabaseReset(request))
        {
         PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- 모든 심볼을 살펴본 후

//--- 트랜잭션 상태
   if(request_error)
     {
      PrintFormat("Table SYMBOLS: failed to add %d symbols", symbols);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table SYMBOLS: added %d symbols",symbols);
     }

//--- SYMBOLS 테이블을 CSV 파일에 저장
   string csv_filename="symbols.csv";
   if(DatabaseExport(db, "SELECT * FROM SYMBOLS", csv_filename,
                     DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|DATABASE_EXPORT_QUOT
      Print("데이터베이스: 테이블 SYMBOLS 저장 위치 ", csv_filename);
   else
      Print("데이터베이스: DatabaseExport(₩"SELECT * FROM SYMBOLS\") 가 코드로 실패", Get
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 데이터베이스 파일을 닫고 이를 알림
   DatabaseClose(db);
   PrintFormat("데이터베이스: %s 생성되고 마감됨", filename);
  }
//+------------------------------------------------------------------+
//| 심볼 사양은 JSON으로 반환                                        |
//+------------------------------------------------------------------+
string GetSymBolAsJson(string symbol)
  {
//--- 들여쓰기
   string indent1=Indent(1);
   string indent2=Indent(2);
   string indent3=Indent(3);
//---
   int digits=(int)SymbolInfoInteger(symbol, SYMBOL_DIGITS);
   string json="{"+
               "\n"+indent1+"\"ConfigSymbols\":["+
               "\n"+indent2+"{"+
               "\n"+indent3+"\"Symbol\":\""+symbol+"\","+
               "\n"+indent3+"\"Path\":\""+SymbolInfoString(symbol, SYMBOL_PATH)+"\","+
               "\n"+indent3+"\"CurrencyBase\":\""+SymbolInfoString(symbol, SYMBOL_CURR
               "\n"+indent3+"\"CurrencyProfit\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"CurrencyMargin\":\""+SymbolInfoString(symbol, SYMBOL_CU
               "\n"+indent3+"\"ColorBackground\":\""+ColorToString((color)SymbolInfoIn
               "\n"+indent3+"\"Digits\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"Point\":\""+DoubleToString(SymbolInfoDouble(symbol, SYM
               "\n"+indent3+"\"TickBookDepth\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ChartMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"TradeCalcMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"OrderMode\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"CalculationMode\":\""+IntegerToString(SymbolInfoIntege
               "\n"+indent3+"\"ExecutionMode\":\""+IntegerToString(SymbolInfoInteger(s
               "\n"+indent3+"\"ExpirationMode\":\""+IntegerToString(SymbolInfoInteger
               "\n"+indent3+"\"FillFlags\":\""+IntegerToString(SymbolInfoInteger(symbo
               "\n"+indent3+"\"ExpirFlags\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"Spread\":\""+IntegerToString(SymbolInfoInteger(symbol,
               "\n"+indent3+"\"TickValue\":\""+StringFormat("%G", (SymbolInfoDouble(sy
               "\n"+indent3+"\"TickSize\":\""+StringFormat("%G", (SymbolInfoDouble(syr
               "\n"+indent3+"\"ContractSize\":\""+StringFormat("%G",(SymbolInfoDouble
               "\n"+indent3+"\"StopsLevel\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"VolumeMin\":\""+StringFormat("%G",(SymbolInfoDouble(syr
               "\n"+indent3+"\"VolumeMax\":\""+StringFormat("%G",(SymbolInfoDouble(syr
               "\n"+indent3+"\"VolumeStep\":\""+StringFormat("%G",(SymbolInfoDouble(sy
               "\n"+indent3+"\"VolumeLimit\":\""+StringFormat("%G",(SymbolInfoDouble(s
               "\n"+indent3+"\"SwapMode\":\""+IntegerToString(SymbolInfoInteger(symbol
               "\n"+indent3+"\"SwapLong\":\""+StringFormat("%G",(SymbolInfoDouble(symb
               "\n"+indent3+"\"SwapShort\":\""+StringFormat("%G",(SymbolInfoDouble(syr
```

© 2000-2025, MetaQuotes Ltd.

```
                "\n"+indent3+"\"Swap3Day\":\""+IntegerToString(SymbolInfoInteger(symbol
                "\n"+indent2+"}"+
                "\n"+indent1+"]"+
                "\n}";
   return(json);
  }
//+------------------------------------------------------------------+
//| 공백으로 된 들여쓰기 구성                                        |
//+------------------------------------------------------------------+
string Indent(const int number, const int characters=3)
  {
   int length=number*characters;
   string indent=NULL;
   StringInit(indent, length, ' ');
   return indent;
  }
/*
 결과:
 데이터베이스: symbols.sqlite 성공적으로 열림
  #| cid name        type notnull dflt_value pk
  -+----------------------------------------
  1|   0 NAME        TEXT       1             0
  2|   1 DESCRIPTION TEXT       0             0
  3|   2 PATH        TEXT       0             0
  4|   3 SPREAD      INT        0             0
  5|   4 POINT       REAL       1             0
  6|   5 DIGITS      INT        1             0
  7|   6 JSON        BLOB       0             0
 1: EURUSD 추가됨
 2: GBPUSD 추가됨
 3: USDCHF 추가됨
  ...
 82: USDCOP 추가됨
 83: USDARS 추가됨
 84: USDCLP 추가됨
 테이블 SYMBOLS: 심볼 84개 추가
 데이터베이스: 테이블 SYMBOLS이 symbols.csv에 저장됨
 데이터베이스: symbols.sqlite 생성되고 닫힘
*/
```

추가 참조

DatabasePrepare, DatabaseBind, DatabaseBindArray, DatabaseFinalize

© 2000-2025, MetaQuotes Ltd.

DatabaseBind

요청에 매개변수 값을 설정.

```
bool  DatabaseBind(
   int   request,       // DatabasePrepare에서 생성된 요청의 핸들
   int   index,         // 요청의 매개변수 인덱스
   T     value          // 단순 유형 매개변수의 값
   );
```

매개변수

request

[in] DatabasePrepare()에서 생성된 요청 핸들.

index

[in] 요청에서 값을 설정해야 하는 매개변수 인덱스. 번호는 0으로 시작합니다.

value

[in] 설정될 값. 확장 유형: bool, char, uchar, short, ushart, int, uint, color, datetime, long, ulong, float, double, string.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INVALID_PARAMETER (4003) - 지원되지 않는 유형;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_NOT_READY (5128) - 현재 기능을 사용하여 요청할 수 없습니다. 요청이 실행 중이거나 이미 완료되었습니다. DatabaseReset()을 호출해야 합니다.

참고

함수는 SQL 요청에 "?" 또는 "?N" 가 포함된 경우에 사용되며, 매개변수 가능 값, 여기서 N은 매개변수 인덱스(하나의 값에서 시작)를 의미합니다. 동시에 DatabaseBind()의 파라미터 인덱싱은 0부터 시작합니다.

예를 들어:

```
INSERT INTO table VALUES (?,?,?)
SELECT * FROM table WHERE id=?
```

DatabasePrepare()에서 매개변수화된 요청이 생성된 후 또는 DatabaseReset()을 사용하여 요청을 재설정한 후 즉시 함수를 호출할 수 있습니다.

이 기능을DatabaseReset()과 함께 사용하면 필요한 횟수만큼 다른 매개변수 값으로 요청을 실행할 수 있습니다.

함수는 단순 유형 매개변수로 작동하도록 고안되었습니다. 배열에 대해 매개변수를 확인해야하는 경우 DatabaseBindArray() 기능을 사용합니다.

예:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlTick ticks[];
//--- 틱을 받기 전에 시작 시간을 기억
   uint start=GetTickCount();
//--- 일별 틱을 요청
   ulong to=TimeCurrent()*1000;
   ulong from=to-PeriodSeconds(PERIOD_D1)*1000;
   if(CopyTicksRange(_Symbol, ticks, COPY_TICKS_ALL, from, to)==-1)
     {
      PrintFormat("%s: CopyTicksRange(%s - %s) failed, error=%d",
                  _Symbol, TimeToString(datetime(from/1000)), TimeToString(datetime(to
      return;
     }
   else
     {
      //--- 얼마나 많은 틱이 수신되었고 얼마나 많은 시간이 걸렸는지
      PrintFormat("%s: CopyTicksRange received %d ticks in %d ms (from %s to %s)",
                  _Symbol, ArraySize(ticks), GetTickCount()-start,
                  TimeToString(datetime(from/1000)), TimeToString(datetime(to/1000)));
     }

//--- 데이터베이스 저장을 위한 파일 이름 설정
   string filename=_Symbol+" "+TimeToString(datetime(from/1000))+" - "+TimeToString(da
   StringReplace(filename, ":", "."); // ":" character is not allowed in file names
//--- 공용 터미널 폴더에서 데이터베이스 열기/생성
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("데이터 베이스: ", filename, " 코드로 여는데 실패 ", GetLastError());
      return;
     }
   else
      Print("데이터베이스: ", filename, " 성공적으로 열림");

//--- TICKS 테이블 생성
   if(!DatabaseExecute(db, "CREATE TABLE TICKS("
                       "SYMBOL             CHAR(10),"
                       "TIME               INT NOT NULL,"
                       "BID                REAL,"
                       "ASK                REAL,"
                       "LAST               REAL,"
                       "VOLUME             INT,"
                       "TIME_MSC           INT,"
                       "VOLUME_REAL        REAL);"))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      Print("DB: ", filename, " create table TICKS failed with code ", GetLastError())
      DatabaseClose(db);
      return;
     }
//--- TICKS 테이블에 모든 필드 목록 표시
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(TICKS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(TICKS)\") failed, error code=%d a
      DatabaseClose(db);
      return;
     }
//--- 매개변수화된 요청을 생성하여 TICKS 테이블에 틱 추가
   string sql="INSERT INTO TICKS (SYMBOL,TIME,BID,ASK,LAST,VOLUME,TIME_MSC,VOLUME_REAL
              " VALUES (?1,?2,?3,?4,?5,?6,?7,?8)"; // 매개변수 요청
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL 요청: ", sql);
      DatabaseClose(db);
      return;
     }
//--- 첫 번째 요청 매개변수의 값을 설정
   DatabaseBind(request, 0, _Symbol);
//--- TICKS 테이블에 틱 추가 전 시작 시간을 기억table
   start=GetTickCount();
   DatabaseTransactionBegin(db);
   int total=ArraySize(ticks);
   bool request_error=false;
   for(int i=0; i<total; i++)
     {
      //--- 항목을 추가하기 전에 나머지 매개변수의 값을 설정
      ResetLastError();
      if(!DatabaseBind(request, 1, ticks[i].time))
        {
         PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
         PrintFormat("Tick #%d line=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
      //--- 이전 DatabaseBind() 호출에 성공한 경우 다음 매개변수를 설정합니다
      if(!request_error && !DatabaseBind(request, 2, ticks[i].bid))
        {
         PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
         PrintFormat("Tick #%d line=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
      if(!request_error && !DatabaseBind(request, 3, ticks[i].ask))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 4, ticks[i].last))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 5, ticks[i].volume))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 6, ticks[i].time_msc))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 7, ticks[i].volume_real))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }

    //--- 진입 입력 요청을 실행하고 오류를 확인
    if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_
      {
       PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
    //--- 다음 매개변수 업데이트 전에 요청을 재설정
    if(!request_error && !DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
```

© 2000-2025, MetaQuotes Ltd.

```
         break;
        }
     } //--- 모든 틱을 다 거쳐 완료

//--- 트랜잭션 상태
   if(request_error)
     {
      PrintFormat("Table TICKS: failed to add %d ticks ", ArraySize(ticks));
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table TICKS: added %d ticks in %d ms",
                  ArraySize(ticks), GetTickCount()-start);
     }

//--- 데이터베이스 파일을 닫고 이를 알림
   DatabaseClose(db);
   PrintFormat("데이터베이스: %s 생성되고 마감됨", filename);
  }
/*
 결과:
 EURUSD: CopyTicksRange는 268061틱을 47 ms 내 수신 (2020.03.18 12:40부터 2020.03.19 12:4
 데이터베이스: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite 열기 성공
 #| cid 이름         유형      notnull dflt_value pk
  -+-----------------------------------------------
  1|   0 SYMBOL      CHAR(10)        0              0
  2|   1 TIME        INT             1              0
  3|   2 BID         REAL            0              0
  4|   3 ASK         REAL            0              0
  5|   4 LAST        REAL            0              0
  6|   5 VOLUME      INT             0              0
  7|   6 TIME_MSC    INT             0              0
  8|   7 VOLUME_REAL REAL            0              0
 TICKS 표: 268061 틱이 797 ms 내 추가됨
 데이터베이스: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite 생성 및 마감
  OnCalculateCorrelation=0.87 2020.03.19 13:00:  EURUSD vs GBPUSD  PERIOD_M30
*/
```

추가 참조

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBindArray

© 2000-2025, MetaQuotes Ltd.

DatabaseBindArray

배열을 매개변수 값으로 설정.

```
bool  DatabaseBind(
   int   request,       // DatabasePrepare에서 생성된 요청의 핸들
   int   index,         // 요청의 매개변수 인덱스
   T&    array[]        // 배열로서 매개변수의 값
   );
```

매개변수

request

[in] DatabasePrepare()에서 생성된 요청 핸들.

index

[in] 요청에서 값을 설정해야 하는 매개변수 인덱스. 번호는 0으로 시작합니다.

array[]

[in] 요청 매개변수 값으로 설정할 배열.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INVALID_PARAMETER (4003) - 지원되지 않는 유형;
- ERR_ARRAY_BAD_SIZE (4011) - 배열 크기(바이트)가 INT_MAX를 초과합니다;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_NOT_READY (5128) - 현재 이 기능을 사용하여 요청을 만들 수 없습니다(요청이 실행 중이거나 이미 완료된 경우, DatabaseReset을 호출해야 함).

참고

함수는 SQL 요청에 "?" 또는 "?N" 가 포함된 경우에 사용되며, 매개변수 가능 값, 여기서 N은 매개변수 인덱스(하나의 값에서 시작)를 의미합니다. 동시에 DatabaseBindArray()에서 매개변수 인덱싱이 0에서 시작됩니다.

예를 들어:

```
INSERT INTO table VALUES (?,?,?)
```

DatabasePrepare()에서 매개변수화된 요청이 생성된 후 또는 DatabaseReset()을 사용하여 요청을 재설정한 후 즉시 함수를 호출할 수 있습니다.

이 기능을DatabaseReset()과 함께 사용하면 필요한 횟수만큼 다른 매개변수 값으로 요청을 실행할 수 있습니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- DAT 확장자를 사용하여 파일을 선택할 수 있는 대화 상자 열기
   string selected_files[];
   if(!FileSelectDialog("다운로드할 파일을 선택", NULL,
                        "Data files (*.dat)|*.dat|All files (*.*)|*.*",
                        FSD_ALLOW_MULTISELECT, selected_files, "tester.dat")>0)
     {
      Print("파일이 선택되지 않음. Exit");
      return;
     }
//--- 파일의 사이즈를 가져오기
   ulong filesize[];
   int filehandle[];
   int files=ArraySize(selected_files);
   ArrayResize(filesize, files);
   ZeroMemory(filesize);
   ArrayResize(filehandle, files);
   double total_size=0;
   for(int i=0; i<files; i++)
     {
      filehandle[i]=FileOpen(selected_files[i], FILE_READ|FILE_BIN);
      if(filehandle[i]!=INVALID_HANDLE)
        {
         filesize[i]=FileSize(filehandle[i]);
         //PrintFormat("%d, %s handle=%d %d bytes", i, selected_files[i], filehandle[i
         total_size+=(double)filesize[i];
        }
     }
//--- 파일의 공통 크기를 확인
   if(total_size==0)
     {
      PrintFormat("총 파일의 크기가 0입니다. Exit");
      return;
     }

//--- 공통 터미널 폴더에서 데이터베이스 생성 또는 열기
   string filename="dat_files.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드와함께 열기 실패 ", GetLastError());
      return;
     }
   else
      Print("데이터베이스: ", filename, " 성공적으로 열림");
//--- FILES 테이블이 있는 경우 삭제
   if(DatabaseTableExists(db, "FILES"))
     {
      //--- 테이블 삭제
      if(!DatabaseExecute(db, "DROP TABLE FILES"))
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          Print("코드가 있는 테이블 FILES을 삭제하는데 실패 ", GetLastError());
          DatabaseClose(db);
          return;
         }
      }
//--- FILES 테이블을 생성
   if(!DatabaseExecute(db, "CREATE TABLE FILES("
                           "NAME           TEXT NOT NULL,"
                           "SIZE           INT  NOT NULL,"
                           "PERCENT_SIZE   REAL NOT NULL,"
                           "DATA           BLOB NOT NULL);"))
      {
       Print("DB: 코드가 있는 FILES을 생성하지 못했습니다 ", GetLastError());
       DatabaseClose(db);
       return;
      }
//--- 코드가 있는 FILES 테이블 파일을 생성하지 못했습니다
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(FILES)", 0)<0)
      {
       PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(FILES)\") failed, error code=%d a
       DatabaseClose(db);
       return;
      }

//--- 매개변수화된 요청을 생성하여 FILES 테이블에 추가
   string sql="INSERT INTO FILES (NAME,SIZE,PERCENT_SIZE,DATA)"
              " VALUES (?1,?2,?3,?4);"; // 매개변수 요청
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
      {
       PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
       Print("SQL 요청: ", sql);
       DatabaseClose(db);
       return;
      }

//--- 모든 파일을 살펴본 후 FILES 테이블에 추가
   bool request_error=false;
   DatabaseTransactionBegin(db);
   int count=0;
   uint size;
   for(int i=0; i<files; i++)
      {
       if(filehandle[i]!=INVALID_HANDLE)
         {
          char data[];
          size=FileReadArray(filehandle[i], data);
          if(size==0)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       PrintFormat("FileReadArray(%s) failed with code %d", selected_files[i], Ge
       continue;
      }

    count++;
    //--- 테이블에 파일을 추가하기 전에 매개변수의 값을 설정
    if(!DatabaseBind(request, 0, selected_files[i]))
      {
       PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
       request_error=true;
       break;
      }
    if(!DatabaseBind(request, 1, size))
      {
       PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
       request_error=true;
       break;
      }
    if(!DatabaseBind(request, 2, double(size)*100./total_size))
      {
       PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
       request_error=true;
       break;
      }
    if(!DatabaseBindArray(request, 3, data))
      {
       PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
       request_error=true;
       break;
      }
    //--- 진입 입력 요청을 실행하고 오류를 확인
    if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
      {
       PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
    else
       PrintFormat("%d. %s: %d bytes", count, selected_files[i],size);
    //--- 다음 매개변수 업데이트 전에 요청을 재설정
    if(!DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
```

© 2000-2025, MetaQuotes Ltd.

```
         }
      }
//--- 트랜잭션 상태
   if(request_error)
     {
      PrintFormat("테이블 FILES: %d 파일 추가 실패", count);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("테이블 FILES: %d 파일 추가", count);
     }

//--- 데이터베이스 파일을 닫고 이를 알림
   DatabaseClose(db);
   PrintFormat("데이터베이스: %s 생성되고 마감됨", filename);
  }
```

추가 참조

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBind

© 2000-2025, MetaQuotes Ltd.

DatabaseRead

요청의 결과로 다음 항목으로 이동.

```
bool  DatabaseRead(
   int  request      // DatabasePrepare에서 요청 핸들 수신
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INVALID_PARAMETER (4003) - 테이블 이름이 지정되지 않음(빈 문자열 또는 NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - 요청을 UTF-8 문자열로 변환 중에 오류 발생;
- ERR_DATABASE_INTERNAL (5120) - 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_EXECUTE (5124) - 요청 실행 오류;
- ERR_DATABASE_NO_MORE_DATA (5126) - 테이블이 없음 (오류 없음, 정상 완료).

추가 참조

DatabasePrepare, DatabaseReadBind

© 2000-2025, MetaQuotes Ltd.

DatabaseReadBind

다음 기록으로 이동하여 기록에서 구조로 데이터를 읽기.

```
bool  DatabaseReadBind(
   int     request,           // DatabasePrepare에서 생성된 요청 핸들
   void&   struct_object      // 기록을 판단하기 위한 구조에 대한 참조
   );
```

매개변수

request

[in] DatabasePrepare()에서 생성된 요청 핸들.

struct_object

[out] 현재 기록의 데이터를 읽을 구조에 대한 참조. 구조에는 멤버로 숫자 유형 및/또는 문자열(배열은 허용되지 않음)만 있어야 하며 하위 항목은 될 수 없습니다.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INVALID_PARAMETER (4003) - 테이블 이름이 지정되지 않음(빈 문자열 또는 NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - 요청을 UTF-8 문자열로 변환 중에 오류 발생;
- ERR_DATABASE_INTERNAL (5120) - 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_EXECUTE (5124) - 요청 실행 오류;
- ERR_DATABASE_NO_MORE_DATA (5126) - 테이블이 없음 (오류 없음, 정상 완료).

참고

struct_object 구조의 필드 수가 DatabaseColumnsCount()를 초과하면 안됩니다. *struct_object* 구조의 필드 수가 레코드의 필드 수보다 작으면 부분 판독이 수행됩니다. 나머지 데이터는 해당 DatabaseColumnText(), DatabaseColumnInteger() 및 기타 기능을 사용하여 명시적으로 가져올 수 있습니다.

예:

```
struct Person
  {
   int               id;
   string            name;
   int               age;
   string            address;
   double            salary;
  };
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   int db;
   string filename="company.sqlite";
//--- 열기
   db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATABASE_
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드와 함께 열기 실패 ", GetLastError());
      return;
     }
//--- 테이블 COMPANY가 존재하면 테이블을 놓습니다
   if(DatabaseTableExists(db, "COMPANY"))
     {
      //--- 테이블 삭제
      if(!DatabaseExecute(db, "DROP TABLE COMPANY"))
        {
         Print("코드가 있는 테이블 COMPANY 삭제 실패 ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- 테이블 생성
   if(!DatabaseExecute(db, "CREATE TABLE COMPANY("
                       "ID INT PRIMARY KEY     NOT NULL,"
                       "NAME           TEXT    NOT NULL,"
                       "AGE            INT     NOT NULL,"
                       "ADDRESS        CHAR(50),"
                       "SALARY         REAL );"))
     {
      Print("DB: ", filename, " 코드로 테이블 생성 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- 데이터 삽입
   if(!DatabaseExecute(db, "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (1
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (2, 'A
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (3, 'T
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (4, 'M
     {
      Print("DB: ", filename, " 코드로 삽입하는데 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }

//--- 요청 준비
   int request=DatabasePrepare(db, "SELECT * FROM COMPANY WHERE SALARY>15000");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드를 이용한 요청 실패 ", GetLastError());
```

© 2000-2025, MetaQuotes Ltd.

```
      DatabaseClose(db);
      return;
     }
//--- 기록 인쇄
   Person person;
   Print("급여 인원 > 15000:");
   for(int i=0; DatabaseReadBind(request, person); i++)
      Print(i, ":  ", person.id, " ", person.name, " ", person.age, " ", person.addres
//--- 사용 후 요청 삭제
   DatabaseFinalize(request);

   Print("일부 통계:");
//--- 총 급여에 대한 새로운 요구를 준비
   request=DatabasePrepare(db, "SELECT SUM(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드를 이용한 요청 실패 ", GetLastError());
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double total_salary;
      DatabaseColumnDouble(request, 0, total_salary);
      Print("총 급여=", total_salary);
     }
//--- 사용 후 요청 삭제
   DatabaseFinalize(request);

//--- 평균 급여에 대한 새로운 요구를 준비
   request=DatabasePrepare(db, "SELECT AVG(SALARY) FROM COMPANY");
   if(request==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드를 이용한 요청 실패 ", GetLastError());
      ResetLastError();
      DatabaseClose(db);
      return;
     }
   while(DatabaseRead(request))
     {
      double aver_salary;
      DatabaseColumnDouble(request, 0, aver_salary);
      Print("평균 급여=", aver_salary);
     }
//--- 사용 후 요청 삭제
   DatabaseFinalize(request);

//--- 데이터베이스 닫기
   DatabaseClose(db);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
/*
Output:
Persons with salary > 15000:
0:  1 Paul 32 California 25000.0
1:  3 Teddy 23 Norway 20000.0
2:  4 Mark 25 Rich-Mond  65000.0
Some statistics:
Total salary=125000.0
Average salary=31250.0
*/
```

추가 참조

DatabasePrepare, DatabaseRead

© 2000-2025, MetaQuotes Ltd.

DatabaseFinalize

DatabasePrepare()에서 생성된 요청을 제거.

```
void  DatabaseFinalize(
   int  request      // DatabasePrepare에서 요청 핸들 수신
   );
```

매개변수

request

[in] DatabasePrepare()에서 요청 핸들 수신.

반환 값

없음.

참고

핸들이 잘못된 경우 함수는 ERR_DATABASE_INVALID_HANDLE 오류를 설정합니다. GetLastError()를 사용하여 오류를 확인할 수 있습니다.

추가 참조

DatabasePrepare, DatabaseExecute

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionBegin

트랜잭션 실행 시작.

```
bool  DatabaseTransactionBegin(
   int  database      // DatabaseOpen에서 수신된 데이터베이스 핸들
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INTERNAL_ERROR (4001) – 심각한 런타임 오류;
- ERR_INVALID_PARAMETER (4003) – sql 매개변수의 문자열이 비어있음;
- ERR_NOT_ENOUGH_MEMORY (4004) – 충분하지 않은 메모리;
- ERR_WRONG_STRING_PARAMETER (5040) – 요청을 UTF-8 문자열로 변환 중에 오류 발생;
- ERR_DATABASE_INTERNAL (5120) – 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) – 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_EXECUTE (5124) – 요청 실행 오류.

참고

트랜잭션을 실행하기 전에 DatabaseTransactionBegin() 기능을 호출해야 합니다. 모든 트랜잭션은 DatabaseTransactionBegin() 호출을 시작하고 DatabaseTransactionCommit() 호출로 끝나야 합니다.

예:

```
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 파일 이름 생성
   string filename=AccountInfoString(ACCOUNT_SERVER) +"_"+IntegerToString(AccountInfo
//--- 공용 터미널 폴더에서 데이터베이스 열기/생성
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " 코드와 함께 열기 실패 ", GetLastError());
      return;
     }
//--- DEALS 표가 이미 있는 경우 삭제
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
//--- DEALS 표를 작성
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }
//---  전체 트레이딩 내역을 요청
   datetime from_date=0;
   datetime to_date=TimeCurrent();
//--- 거래 내역을 지정 간격 내에 요청
   HistorySelect(from_date, to_date);
   int deals_total=HistoryDealsTotal();
   PrintFormat("트레이딩 내역 내 거래: %d ", deals_total);

//--- DatabaseTransactionBegin/DatabaseTransactionCommit을 사용하여 트랜잭션 실행 속도를 측
   ulong start=GetMicrosecondCount();
   bool fast_transactions=true;
   InsertDeals(db, fast_transactions);
   double fast_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit

//--- DEALS 테이블을 삭제하고 다시 생성
   if(!DeleteTable(db, "DEALS"))
     {
      DatabaseClose(db);
      return;
     }
//--- 새 DEALS 테이블 생성
   if(!CreateTableDeals(db))
     {
      DatabaseClose(db);
      return;
     }

//--- 이번에는 DatabaseTransactionBegin/DatabaseTransactionCommit를 사용하지 않고 다시 테스
   fast_transactions=false;
   start=GetMicrosecondCount();
   InsertDeals(db, fast_transactions);
   double slow_transactions_time=double(GetMicrosecondCount()-start)/1000;
   PrintFormat("Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit
//--- 시간의 증가를 보고
   PrintFormat("DatabaseTransactionBegin/DatabaseTransactionCommit 를 사용하여 %.1f 배
//--- 데이터베이스 닫기
   DatabaseClose(db);
  }
/*
Results:
   Deals in the trading history: 2737
   Transations WITH    DatabaseTransactionBegin/DatabaseTransactionCommit: time=48.5 r
   Transations WITHOUT DatabaseTransactionBegin/DatabaseTransactionCommit: time=25818.
```

© 2000-2025, MetaQuotes Ltd.

```
   Use of DatabaseTransactionBegin/DatabaseTransactionCommit provided acceleration by
*/
//+------------------------------------------------------------------+
//| 데이터베이스에서 지정된 이름의 표를 삭제                         |
//+------------------------------------------------------------------+
bool DeleteTable(int database, string table_name)
  {
   if(!DatabaseExecute(database, "DROP TABLE IF EXISTS "+table_name))
     {
      Print("코드가 있는 테이블을 삭제 실패 ", GetLastError());
      return(false);
     }
//--- 테이블이 성공적으로 삭제되었습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| DEALS 표 생성                                                    |
//+------------------------------------------------------------------+
bool CreateTableDeals(int database)
  {
//--- 표의 존재유무 확인
   if(!DatabaseTableExists(database, "DEALS"))
      //--- 표 생성
      if(!DatabaseExecute(database, "CREATE TABLE DEALS("
                          "ID          INT KEY NOT NULL,"
                          "ORDER_ID    INT     NOT NULL,"
                          "POSITION_ID INT     NOT NULL,"
                          "TIME        INT     NOT NULL,"
                          "TYPE        INT     NOT NULL,"
                          "ENTRY       INT     NOT NULL,"
                          "SYMBOL      CHAR(10),"
                          "VOLUME      REAL,"
                          "PRICE       REAL,"
                          "PROFIT      REAL,"
                          "SWAP        REAL,"
                          "COMMISSION  REAL,"
                          "MAGIC       INT,"
                          "REASON      INT );"))
        {
         Print("DB: 코드로 실패한 DEALS 테이블을 작성 ", GetLastError());
         return(false);
        }
//--- 표가 성공적으로 생성되었습니다
   return(true);
  }
//+------------------------------------------------------------------+
//| 데이터베이스 표에 거래 추가                                      |
//+------------------------------------------------------------------+
bool InsertDeals(int database, bool begintransaction=true)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
//--- 보조 변수
   ulong    deal_ticket;          // 거래 티켓
   long     order_ticket;         // 거래가 실행된 주문 티켓
   long     position_ticket;      // 거래가 속한 포지션의 ID
   datetime time;                 // 거래 실행 시간
   long     type ;                // 거래 유형
   long     entry ;               // 거래 방향
   string   symbol;               // 거래가 실행된 심볼
   double   volume;               // 작업 볼륨
   double   price;                // 가격
   double   profit;               // 재정적 결과
   double   swap;                 // 스왑
   double   commission;           // 수수료
   long     magic;                // 매직 넘버
   long     reason;               // 거래 실행 이유 또는 소스
//--- 모든 거래를 검토하고 데이터베이스에 추가
   bool failed=false;
   int deals=HistoryDealsTotal();
//--- 빠른 트랜잭션 성능 방법을 사용하는 경우
   if(begintransaction)
     {
      // --- 트랜잭션을 실행하기 전에 데이터베이스를 잠금
      DatabaseTransactionBegin(database);
     }
   for(int i=0; i<deals; i++)
     {
      deal_ticket=    HistoryDealGetTicket(i);
      order_ticket=   HistoryDealGetInteger(deal_ticket, DEAL_ORDER);
      position_ticket=HistoryDealGetInteger(deal_ticket, DEAL_POSITION_ID);
      time= (datetime)HistoryDealGetInteger(deal_ticket, DEAL_TIME);
      type=           HistoryDealGetInteger(deal_ticket, DEAL_TYPE);
      entry=          HistoryDealGetInteger(deal_ticket, DEAL_ENTRY);
      symbol=         HistoryDealGetString(deal_ticket, DEAL_SYMBOL);
      volume=         HistoryDealGetDouble(deal_ticket, DEAL_VOLUME);
      price=          HistoryDealGetDouble(deal_ticket, DEAL_PRICE);
      profit=         HistoryDealGetDouble(deal_ticket, DEAL_PROFIT);
      swap=           HistoryDealGetDouble(deal_ticket, DEAL_SWAP);
      commission=     HistoryDealGetDouble(deal_ticket, DEAL_COMMISSION);
      magic=          HistoryDealGetInteger(deal_ticket, DEAL_MAGIC);
      reason=         HistoryDealGetInteger(deal_ticket, DEAL_REASON);
      //--- 다음 요청에 따ㅏ 각 거래를 추가
      string request_text=StringFormat("INSERT INTO DEALS (ID,ORDER_ID,POSITION_ID,TIN
                                       "VALUES (%d, %d, %d, %d, %d, %d, '%s', %G, %G,
                                       deal_ticket, order_ticket, position_ticket, tir
      if(!DatabaseExecute(database, request_text))
        {
         PrintFormat("%s: failed to insert deal #%dwith code %d", __FUNCTION__, deal_t
         PrintFormat("i=%d: deal #%d  %s", i, deal_ticket, symbol);
```

© 2000-2025, MetaQuotes Ltd.

```
          failed=true;
          break;
         }
      }
//--- 트랜잭션 실행 오류를 확인
   if(failed)
     {
      //--- 빠른 트랜잭션 성능 방법을 사용하는 경우
      if(begintransaction)
        {
         //--- 모든 트랜잭션 롤백 및 데이터베이스 잠금 해제
         DatabaseTransactionRollback(database);
        }
      Print("%s: DatabaseExecute() failed with code ", __FUNCTION__, GetLastError());
      return(false);
     }
//--- 빠른 트랜잭션 성능 방법을 사용하는 경우
   if(begintransaction)
     {
      //--- 모든 트랜잭션이 성공적으로 수행됨 - 변경사항을 기록하고 데이터베이스 잠금을 해제
      DatabaseTransactionCommit(database);
     }
//--- 성공적으로 완료
   return(true);
  }
//+------------------------------------------------------------------+
```

추가 참조

DatabaseExecute, DatabasePrepare, DatabaseTransactionCommit, DatabaseTransactionRollback

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionCommit

트랜잭션 실행을 완료.

```
bool  DatabaseTransactionCommit(
   int  database      // DatabaseOpen에서 수신된 데이터베이스 핸들
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INTERNAL_ERROR (4001) - 심각한 런타임 오류;
- ERR_INVALID_PARAMETER (4003) - sql 매개변수의 문자열이 비어있음;
- ERR_NOT_ENOUGH_MEMORY (4004) - 충분하지 않은 메모리;
- ERR_WRONG_STRING_PARAMETER (5040) - 요청을 UTF-8 문자열로 변환 중에 오류 발생;
- ERR_DATABASE_INTERNAL (5120) - 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_EXECUTE (5124) - 요청 실행 오류.

참고

DatabaseTransactionCommit() 함수는 DatabaseBeginTransaction() 함수를 호출한 후 실행된 모든 트랜잭션을 완료합니다. 모든 트랜잭션은 DatabaseTransactionBegin() 호출로 시작하고 성공적으로 완료하기 위해 DatabaseTransactionCommit() 호출로 끝나야 합니다.

추가 참조

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionRollback

© 2000-2025, MetaQuotes Ltd.

DatabaseTransactionRollback

롤백 트랜잭션.

```
bool  DatabaseTransactionRollback(
   int  database      // DatabaseOpen에서 수신된 데이터베이스 핸들
   );
```

매개변수

database

[in] DatabaseOpen()로 수신된 데이터베이스 핸들.

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_INTERNAL_ERROR (4001) - 심각한 런타임 오류;
- ERR_INVALID_PARAMETER (4003) - sql 매개변수가 빈 문자열을 포함;
- ERR_NOT_ENOUGH_MEMORY (4004) - 충분하지 않은 메모리;
- ERR_WRONG_STRING_PARAMETER (5040) - 요청을 UTF-8 문자열로 변환 중에 오류 발생;
- ERR_DATABASE_INTERNAL (5120) - 내부 데이터베이스 오류;
- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 데이터베이스 핸들;
- ERR_DATABASE_EXECUTE (5124) - 요청 실행 오류.

참고

DatabaseTransactionRollback() 호출은 DatabaseTransactionBegin() 함수를 호출한 후 실행된 모든 트랜잭션을 취소합니다. DatabaseTransactionRollback() 기능은 트랜잭션을 실행할 때 오류가 발생할 경우 데이터베이스의 변경 사항을 롤백하는 데 필요합니다.

추가 참조

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionCommit

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnsCount

요청의 필드 수를 가져오기.

```
int  DatabaseColumnsCount(
   int  request      // DatabasePrepare에서 요청 핸들 수신
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

반환 값

필드 수 또는 오류 발생 시 -1. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 요청 핸들.

참고

DatabasePrepare()에서 생성된 요청 필드 수를 가져오기 위해 DatabaseRead()함수를 호출할 필요가 없습니다. 나머지 DatabaseColumnXXX() 기능의 경우, DatabaseRead()를 미리 호출해야합니다.

추가 참조

DatabasePrepare, DatabaseFinalize, DatabaseClose

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnName

인덱스별로 필드명을 가져옵니다.

```
bool  DatabaseColumnName(
   int       request,     // DatabasePrepare에서 수신된 요청 핸들
   int       column,      // 요청에서의 필드 인덱스
   string&   name         // 필드명을 수신하는 변수에 대한 레퍼런스
   );
```

Parameters

request

[in] DatabasePrepare()에서 수신된 요청 핸들.

column

[in] 요청에서의 필드 인덱스. 필드 넘버링은 0부터 시작하며 DatabaseColumnsCount() - 1을 초과할 수 없습니다.

name

[out] 필드명 작성 변수.

반환 값

성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다. 오류 코드를 가져오려면 GetLastError()를 사용하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 잘못된 요청 핸들;
- ERR_DATABASE_NO_MORE_DATA (5126) - '칼럼' 인덱스가 DatabaseColumnsCount() -1을 초과함.

참고

이 값은 '요청'을 위해 예비적으로 하나 이상의 DatabaseRead() 요청을 한 경우에만 얻을 수 있습니다.

더 보기

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnType

인덱스별 필드 유형을 가져오기.

```
ENUM_DATABASE_FIELD_TYPE  DatabaseColumnType(
   int  request,     // DatabasePrepare에서 요청 핸들 수신
   int  column       // 요청 필드 인덱스
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

column

[in] 요청 필드 인덱스. 필드 번호 지정은 0부터이며 DatabaseColumnsCount() - 1을 초과할 수 없습니다.

반환 값

필드 유형을 ENUM_DATABASE_FIELD_TYPE 열거 값에서 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 요청 핸들;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' 인덱스는 DatabaseColumnsCount() -1을 초과합니다.

참고

이 값은 '요청'에 대해 하나 이상의 DatabaseRead() 호출을 수행한 경우에만 얻을 수 있습니다.

ENUM_DATABASE_FIELD_TYPE

ID	설명
DATABASE_FIELD_TYPE_INVALID	유형을 가져오는 동안 오류가 발생, 오류 코드는 int GetLastError()를 사용하여 가져올 수 있습니다
DATABASE_FIELD_TYPE_INTEGER	정수 유형
DATABASE_FIELD_TYPE_FLOAT	실수 유형
DATABASE_FIELD_TYPE_TEXT	문자열 유형
DATABASE_FIELD_TYPE_BLOB	바이너리 유형
DATABASE_FIELD_TYPE_NULL	특수 NULL 유형

추가 참조

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnSize

필드 크기(바이트)를 가져오기.

```
int  DatabaseColumnSize(
   int  request,     // DatabasePrepare에서 요청 핸들 수신
   int  column       // 요청 필드 인덱스
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

column

[in] 요청 필드 인덱스. 필드 번호 지정은 0부터이며 DatabaseColumnsCount() - 1을 초과할 수 없습니다.

반환 값

성공하면 필드 크기(바이트)가, 그렇지 않으면 -1이 반환됩니다. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 요청 핸들;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' 인덱스가 DatabaseColumnsCount() -1을 초과합니다.

참고

이 값은 '요청'에 대해 하나 이상의 DatabaseRead() 호출을 수행한 경우에만 얻을 수 있습니다.

추가 참조

DatabasePrepare, DatabaseColumnBlob, DatabaseColumnsCount, DatabaseColumnName, DatabaseColumnType

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnText

현재 기록에서 문자열로 필드 값을 가져오기.

```
bool  DatabaseColumnText(
   int       request,      // DatabasePrepare에서 요청 핸들 수신
   int       column,       // 요청 필드 인덱스
   string&   value         // 값을 수신하기 위한 변수에 대한 참조
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

column

[in] 요청 필드 인덱스. 필드 번호 지정은 0부터이며 DatabaseColumnsCount() - 1을 초과할 수 없습니다.

value

[out] 필드 값 쓰기에 대한 변수 참조.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 요청 핸들;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' 인덱스가 DatabaseColumnsCount() -1을 초과합니다.

참고

이 값은 '요청'에 대해 하나 이상의 DatabaseRead() 호출을 수행한 경우에만 얻을 수 있습니다.

다음 기록에서 값을 값을 읽으려면, 사전에 DatabaseRead()를 호출.

추가 참조

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnInteger

현재 기록에서 int 유형 값을 가져오기.

```
bool  DatabaseColumnInteger(
   int   request,     // DatabasePrepare에서 요청 핸들 수신
   int   column,      // 요청 필드 인덱스
   int&  value        // 값을 수신하기 위한 변수에 대한 참조
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

column

[in] 요청 필드 인덱스. 필드 번호 지정은 0부터이며 DatabaseColumnsCount() - 1을 초과할 수 없습니다.

value

[out] 필드 값 쓰기에 대한 변수 참조.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 요청 핸들;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' 인덱스가 DatabaseColumnsCount() -1을 초과합니다.

참고

이 값은 '요청'에 대해 하나 이상의 DatabaseRead() 호출을 수행한 경우에만 얻을 수 있습니다.

다음 기록에서 값을 값을 읽으려면, 사전에 DatabaseRead()를 호출.

추가 참조

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnLong

현재 기록에서 long 유형 값을 가져오기.

```
bool  DatabaseColumnLong(
   int     request,      // DatabasePrepare 준비에서 요청 핸들 수신
   int     column,       // 요청의 필드 인덱스
   long&   value         // 값을 수신하기 위한 변수에 대한 참조
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

column

[in] 요청 필드 인덱스. 필드 번호 지정은 0부터이며 DatabaseColumnsCount() - 1을 초과할 수 없습니다.

value

[out] 필드 값 쓰기에 대한 변수 참조.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 요청 핸들;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' 인덱스가 DatabaseColumnsCount() -1을 초과합니다.

참고

이 값은 '요청'에 대해 하나 이상의 DatabaseRead() 호출을 수행한 경우에만 얻을 수 있습니다.

다음 기록에서 값을 값을 읽으려면, 사전에 DatabaseRead()를 호출.

추가 참조

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnDouble

현재 레코드에서 double 유형 값을 가져오기.

```
bool  DatabaseColumnDouble(
   int       request,      // DatabasePrepare에서 요청 핸들 수신
   int       column,       // 요청 필드 인덱스
   double&   value         // 값을 수신하기 위한 변수에 대한 참조
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

column

[in] 요청 필드 인덱스. 필드 번호 지정은 0부터이며 DatabaseColumnsCount() - 1을 초과할 수 없습니다.

value

[out] 필드 값 쓰기에 대한 변수 참조.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 요청 핸들;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' 인덱스가 DatabaseColumnsCount() -1을 초과합니다.

참고

이 값은 '요청'에 대해 하나 이상의 DatabaseRead() 호출을 수행한 경우에만 얻을 수 있습니다.

다음 기록에서 값을 값을 읽으려면, 사전에 DatabaseRead()를 호출.

추가 참조

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DatabaseColumnBlob

현재 기록에서 배열로 필드값을 가져오기.

```
bool  DatabaseColumnBlob(
   int     request,      // DatabasePrepare 준비에서 요청 핸들 수신
   int     column,       // 요청의 필드 인덱스
   void&   data[]        // 값을 수신하기 위한 변수에 대한 참조
   );
```

매개변수

request

[in] 요청 핸들을 DatabasePrepare()에서 수신.

column

[in] 요청 필드 인덱스. 필드 번호 지정은 0부터이며 DatabaseColumnsCount() - 1을 초과할 수 없습니다.

data[]

[out] 필드 값을 쓰기 위한 배열 참조.

반환 값

성공하면 true를, 그렇지 않으면 false를 반환. 오류 코드를 가져오려면 GetLastError()를 사용해야 하며, 가능한 응답은 다음과 같습니다:

- ERR_DATABASE_INVALID_HANDLE (5121) - 유효하지 않은 요청 핸들;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' 인덱스가 DatabaseColumnsCount() -1을 초과합니다.

참고

이 값은 '요청'에 대해 하나 이상의 DatabaseRead() 호출을 수행한 경우에만 얻을 수 있습니다.

다음 기록에서 값을 값을 읽으려면, 사전에 DatabaseRead()를 호출.

추가 참조

DatabasePrepare, DatabaseColumnSize, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

© 2000-2025, MetaQuotes Ltd.

DirectX 작업

DirectX 11 기능과 셰이더는 가격표에 직접 3D 시각화를 위해 설계되었습니다.

3D 그래픽을 만들려면 필요한 이미지 크기를 가진 그래픽 컨텍스트(DXContextCreate)가 필요합니다. 또한 정점 및 인덱스 버퍼(DXBufferCreate)뿐만 아니라 정점 및 픽셀 셰이더 생성(DXShaderCreate)도 준비해야합니다. 이 정도면 컬러로 그래픽을 표시하기에 충분합니다.

다음 수준의 그래픽에서는 추가 렌더링 매개변수를 셰이더로 전달하기 위한 입력 (DXInputSet)이 필요합니다. 이를 통해 카메라 및 3D 개체 위치를 설정하고 광원을 표현하며, 마우스 및 키보드 제어를 구현할 수 있습니다.

따라서 내장 MQL5 기능을 사용하면 타사 툴 없이도 MetaTrader 5에서 직접 애니메이션 3D 차트를 만들 수 있습니다. 기능이 작동하려면 비디오 카드가 DX 11 및 셰이더 모델 5.0을 지원해야 합니다.

라이브러리 작업을 시작하려면 MetaTrader 5에서 3D 그래픽을 생성하는 방법을 읽어 보십시오.

기능	액션
DXContextCreate	지정된 크기의 프레임을 렌더링하기 위한 그래픽 컨텍스트를 생성
DXContextSetSize	DXContextCreate()에서 작성된 그래픽 컨텍스트의 프레임 크기를 변경
DXContextSetSize	DXContextCreate()에서 생성된 그래픽 컨텍스트의 프레임 크기를 가져오기
DXContextClearColors	렌더링 버퍼에 대해 지정된 색상을 모든 픽셀로 설정
DXContextClearDepth	깊이 버퍼를 비움
DXContextGetColors	그래픽 컨텍스트에서 지정된 크기 및 오프셋의 이미지를 가져오기
DXContextGetDepth	렌더링된 프레임의 깊이 버퍼를 가져오기
DXBufferCreate	데이터 배열을 기반으로 지정된 유형의 버퍼를 생성
DXTextureCreate	전달된 이미지에서 잘라낸 지정된 크기의 직사각형으로 2D 질감 생성
DXInputCreate	셰이더 입력 생성
DXInputSet	셰이더 입력 설정
DXShaderCreate	지정된 유형의 셰이더 생성
DXShaderSetLayout	정점 셰이더의 꼭지점 레이아웃을 설정
DXShaderInputsSet	셰이더 입력 설정
DXShaderTexturesSet	셰이더 질감 설정
DXDraw	DXBufferSet()에서 정점 버퍼 세트의 점들을 렌더링

© 2000-2025, MetaQuotes Ltd.

기능	액션
DXDrawIndexed	DXBufferSet()에서 인덱스 버퍼가 설명하는 그래픽 기본 요소를 렌더링
DXPrimiveTopologySet	DXDrawIndexed()를 사용하여 렌더링할 기본 요소의 유형을 설정
DXBufferSet	현재 렌더링의 버퍼를 설정
DXShaderSet	렌더링을 위한 셰이더 설정
DXHandleType	핸들 유형을 반환
DXRelease	핸들을 해제

© 2000-2025, MetaQuotes Ltd.

DXContextCreate

지정된 크기의 프레임을 렌더링하기 위한 그래픽 컨텍스트를 작성.

```
int  DXContextCreate(
   uint  width,      // 폭(픽셀)
   uint  height      // 높이(픽셀)
   );
```

매개변수

너비

[in] 프레임 폭(픽셀).

높이

[in] 프레임 높이(픽셀)

값 반환

오류가 발생한 경우 생성된 컨텍스트에 대한 핸들 또는INVALID_HANDLE. 오류 코드를 받으려면 GetLastError() 기능을 호출해야 합니다.

주의

DXBufferCreate, DXInputCreate, DXShaderCreate 및 DXTextureCreate 기능을 사용하여 생성된 모든 그래픽 개체는 해당 개체가 생성된 그래픽 컨텍스트에서만 사용할 수 있습니다.

이후에 프레임 크기를 DXContextSetSize()로 변경할 수 있습니다.

더 이상 사용되지 않는 생성된 핸들은 DXRelease() 기능에 의해 명시적으로 해제되어야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXContextSetSize

DXContextCreate()에서 작성된 그래픽 컨텍스트 프레임 크기를 변경.

```
bool  DXContextSetSize(
   int    context,      // 그래픽 컨텍스트 핸들
   uint&  width,        // 폭(픽셀)
   uint&  height        // 높이(픽셀)
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

너비

[in] 프레임 폭(픽셀).

높이

[in] 프레임 높이(픽셀)

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

그래픽 컨텍스트의 프레임 크기는 프레임 렌더링 사이에서만 변경해야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXContextGetSize

DXContextCreate()에서 작성된 그래픽 컨텍스트 프레임 크기를 가져오기.

```
bool  DXContextGetSize(
   int    context,      // 그래픽 컨텍스트 핸들
   uint&  width,        // 폭(픽셀)
   uint&  height        // 높이(픽셀)
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

너비

[out] 프레임 폭(픽셀).

높이

[out] 프레임 높이(픽셀).

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXContextClearColors

렌더링 버퍼의 모든 픽셀에 지정된 색상을 설정.

```
bool  DXContextClearColors(
   int              context,      // 그래픽 컨텍스트 핸들
   const DXVector&  color         // 색상
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

색상

[in] 렌더링 색상.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

DXContextClearColors() 기능은 다음 프레임을 렌더링하기 전에 색상 버퍼를 지우는 데 사용할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

DXContextClearDepth

Clears the depth buffer.

```
bool  DXContextClearDepth(
   int  context      // 그래픽 컨텍스트 핸들
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

DXContextClearDepth() 기능은 다음 프레임을 렌더링 하기 전에 깊이 버퍼를 지우는데 사용할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

DXContextGetColors

그래픽 컨텍스트에서 지정된 크기 및 간격 띄우기의 이미지를 가져오기.

```
bool  DXContextGetColors(
   int    context,                      // 그래픽 컨텍스트 핸들
   uint&  image[],                      // 이미지 픽셀 배열
   int    image_width=WHOLE_ARRAY,      // 이미지 폭(픽셀)
   int    image_height=WHOLE_ARRAY,     // 이미지 높이(픽셀)
   int    image_offset_x=0,             // X 간격 띄우기
   int    image_offset_y=0              // Y 간격 띄우기
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

이미지

[out] *image_width*image_height* 픽셀 배열 ARGB 형.

image_width=WHOLE_ARRAY

[in] 이미지 폭(픽셀).

image_height=WHOLE_ARRAY

[in] 이미지 높이(픽셀).

image_offset_x=0

[in] X 간격 띄우기.

image_offset_y=0

[in] Y 간격 띄우기.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXContextGetDepth

렌더링된 프레임의 깊이 버퍼를 가져오기

```
bool  DXContextGetDepth(
   int     context,      // 그래픽 컨텍스트 핸들
   float&  image[]       // 깊이 값 배열₩
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

이미지

[out] 렌더링된 프레임 깊이 버퍼 값의 배열.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

반환된 버퍼에는 상대 단위(0.0～1.0)로DXContextGetColors()에서 얻을 수 있는 렌더링된 프레임의 각 픽셀 깊이가 포함됩니다.

© 2000-2025, MetaQuotes Ltd.

DXBufferCreate

데이터 배열을 기반으로 지정된 유형의 버퍼를 생성

```
int  DXBufferCreate(
   int                  context,              // 그래픽 컨텍스트 핸들
   ENUM_DX_BUFFER_TYPE  buffer_type,          // 생성된 버퍼 유형
   const void&          data[],               // 버퍼 데이터
   uint                 start=0,              // 초기 인덱스
   uint                 count=WHOLE_ARRAY     // 요소의 수
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

buffer_type

[in] ENUM_DX_BUFFER_TYPE 열거의 버퍼 유형.

data[]

[in] 버퍼 생성을 위한 데이터.

start=0

[in] 버퍼를 생성하는 데 배열 값이 사용되는 배열의 첫 번째 요소의 인덱스. 기본적으로 데이터는 배열 시작부터 가져옵니다.

count=WHOLE_ARRAY

[in] 값의 수. 기본적으로 전체 배열이 사용됩니다 (count=WHOLE_ARRAY).

값 반환

오류가 발생한 경우 생성된 버퍼의 핸들 또는INVALID_HANDLE. 오류 코드를 받으려면 GetLastError() 기능을 호출해야 합니다.

주의

인덱스 버퍼의 경우, 정점 버퍼가 정점을 나타내는 구조 배열을 수신하는 동안 *data[]* 배열은 'uint' 유형이어야 합니다.

더 이상 사용되지 않는 생성된 핸들은 DXRelease() 기능에 의해 명시적으로 해제되어야 합니다.

ENUM_DX_BUFFER_TYPE

ID	값	설명
DX_BUFFER_VERTEX	1	정점 버퍼
DX_BUFFER_INDEX	2	인덱스 버퍼

© 2000-2025, MetaQuotes Ltd.

DXTextureCreate

전달된 이미지에서 잘라낸 지정된 크기의 직사각형으로 2D 텍스처를 생성.

```
int  DXTextureCreate(
   int              context,          // 그래픽 컨텍스트 핸들
   ENUM_DX_FORMAT   format,           // 픽셀 색상 서식
   uint             width,            // 소스 이미지 폭
   uint             height,           // 소스 이미지 높이
   const void&      data[],           // 소스 이미지 픽셀 배열
   uint             data_x,           // 텍스처를 만드는데 사용되는 직사각형 X 좌표
   uint             data_y,           // 텍스처를 만드는데 사용되는 직사각형의 Y 좌표
   uint             data_width,       // 텍스처를 만들기 위한 직사각형 폭
   uint             data_height       // 텍스터를 만들기 위한 직사각형 높이
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

서식

[in] ENUM_DX_FORMAT 열거에서 픽셀 색상 서식을 설정

너비

[in] 텍스처가 기준인 이미지의 폭.

높이

[in] 텍스처가 기준인 이미지의 높이.

데이터

[in] 텍스처가 기준인 이미지의 픽셀 배열.

data_x

[in] 텍스처를 만드는데 사용되는 직사각형의 X 좌표 (X 간격).

data_y

[in] 텍스처를 만드는데 사용되는 직사각형의 Y 좌표 (Y 간격).

data_width

[in] 텍스처를 만드는데 사용되는 직사각형의 폭.

data_height

[in] 텍스처를 만드는데 사용되는 직사각형의 높이.

값 반환

오류 발생 시 텍스처 핸들 또는 INVALID_HANDLE. 오류 코드를 받으려면 GetLastError() 기능을 호출해야 합니다.

주의

더 이상 사용되지 않는 생성된 핸들은 DXRelease() 기능에 의해 명시적으로 해제되어야 합니다.

© 2000-2025, MetaQuotes Ltd.

ENUM_DX_FORMAT

ID	값	Match in DXGI_FORMAT
DX_FORMAT_UNKNOWN	0	DXGI_FORMAT_UNKNOWN
DX_FORMAT_R32G32B32A32_TYPELESS	1	DXGI_FORMAT_R32G32B32A32_TYPELESS
DX_FORMAT_R32G32B32A32_FLOAT	2	DXGI_FORMAT_R32G32B32A32_FLOAT
DX_FORMAT_R32G32B32A32_UINT	3	DXGI_FORMAT_R32G32B32A32_UINT
DX_FORMAT_R32G32B32A32_SINT	4	DXGI_FORMAT_R32G32B32A32_SINT
DX_FORMAT_R32G32B32_TYPELESS	5	DXGI_FORMAT_R32G32B32_TYPELESS
DX_FORMAT_R32G32B32_FLOAT	6	DXGI_FORMAT_R32G32B32_FLOAT
DX_FORMAT_R32G32B32_UINT	7	DXGI_FORMAT_R32G32B32_UINT
DX_FORMAT_R32G32B32_SINT	8	DXGI_FORMAT_R32G32B32_SINT
DX_FORMAT_R16G16B16A16_TYPELESS	9	DXGI_FORMAT_R16G16B16A16_TYPELESS
DX_FORMAT_R16G16B16A16_FLOAT	10	DXGI_FORMAT_R16G16B16A16_FLOAT
DX_FORMAT_R16G16B16A16_UNORM	11	DXGI_FORMAT_R16G16B16A16_UNORM
DX_FORMAT_R16G16B16A16_UINT	12	DXGI_FORMAT_R16G16B16A16_UINT
DX_FORMAT_R16G16B16A16_SNORM	13	DXGI_FORMAT_R16G16B16A16_SNORM
DX_FORMAT_R16G16B16A16_SINT	14	DXGI_FORMAT_R16G16B16A16_SINT
DX_FORMAT_R32G32_TYPELESS	15	DXGI_FORMAT_R32G32_TYPELESS
DX_FORMAT_R32G32_FLOAT	16	DXGI_FORMAT_R32G32_FLOAT
DX_FORMAT_R32G32_UINT	17	DXGI_FORMAT_R32G32_UINT
DX_FORMAT_R32G32_SINT	18	DXGI_FORMAT_R32G32_SINT
DX_FORMAT_R32G8X24_TYPELESS	19	DXGI_FORMAT_R32G8X24_TYPELESS
DX_FORMAT_D32_FLOAT_S8X24_UINT	20	DXGI_FORMAT_D32_FLOAT_S8X24_UINT
DX_FORMAT_R32_FLOAT_X8X24_TYPELESS	21	DXGI_FORMAT_R32_FLOAT_X8X24_TYPELESS
DX_FORMAT_X32_TYPELESS_G8X24_UINT	22	DXGI_FORMAT_X32_TYPELESS_G8X24_UINT
DX_FORMAT_R10G10B10A2_TYPELESS	23	DXGI_FORMAT_R10G10B10A2_TYPELESS
DX_FORMAT_R10G10B10A2_UNORM	24	DXGI_FORMAT_R10G10B10A2_UNORM

© 2000-2025, MetaQuotes Ltd.

ID	값	Match in DXGI_FORMAT
DX_FORMAT_R10G10B10A2_UINT	25	DXGI_FORMAT_R10G10B10A2_UINT
DX_FORMAT_R11G11B10_FLOAT	26	DXGI_FORMAT_R11G11B10_FLOAT
DX_FORMAT_R8G8B8A8_TYPELESS	27	DXGI_FORMAT_R8G8B8A8_TYPELESS
DX_FORMAT_R8G8B8A8_UNORM	28	DXGI_FORMAT_R8G8B8A8_UNORM
DX_FORMAT_R8G8B8A8_UNORM_SRGB	29	DXGI_FORMAT_R8G8B8A8_UNORM_SRGB
DX_FORMAT_R8G8B8A8_UINT	30	DXGI_FORMAT_R8G8B8A8_UINT
DX_FORMAT_R8G8B8A8_SNORM	31	DXGI_FORMAT_R8G8B8A8_SNORM
DX_FORMAT_R8G8B8A8_SINT	32	DXGI_FORMAT_R8G8B8A8_SINT
DX_FORMAT_R16G16_TYPELESS	33	DXGI_FORMAT_R16G16_TYPELESS
DX_FORMAT_R16G16_FLOAT	34	DXGI_FORMAT_R16G16_FLOAT
DX_FORMAT_R16G16_UNORM	35	DXGI_FORMAT_R16G16_UNORM
DX_FORMAT_R16G16_UINT	36	DXGI_FORMAT_R16G16_UINT
DX_FORMAT_R16G16_SNORM	37	DXGI_FORMAT_R16G16_SNORM
DX_FORMAT_R16G16_SINT	38	DXGI_FORMAT_R16G16_SINT
DX_FORMAT_R32_TYPELESS	39	DXGI_FORMAT_R32_TYPELESS
DX_FORMAT_D32_FLOAT	40	DXGI_FORMAT_D32_FLOAT
DX_FORMAT_R32_FLOAT	41	DXGI_FORMAT_R32_FLOAT
DX_FORMAT_R32_UINT	42	DXGI_FORMAT_R32_UINT
DX_FORMAT_R32_SINT	43	DXGI_FORMAT_R32_SINT
DX_FORMAT_R24G8_TYPELESS	44	DXGI_FORMAT_R24G8_TYPELESS
DX_FORMAT_D24_UNORM_S8_UINT	45	DXGI_FORMAT_D24_UNORM_S8_UINT
DX_FORMAT_R24_UNORM_X8_TYPELESS	46	DXGI_FORMAT_R24_UNORM_X8_TYPELESS
DX_FORMAT_X24_TYPELESS_G8_UINT	47	DXGI_FORMAT_X24_TYPELESS_G8_UINT
DX_FORMAT_R8G8_TYPELESS	48	DXGI_FORMAT_R8G8_TYPELESS
DX_FORMAT_R8G8_UNORM	49	DXGI_FORMAT_R8G8_UNORM
DX_FORMAT_R8G8_UINT	50	DXGI_FORMAT_R8G8_UINT
DX_FORMAT_R8G8_SNORM	51	DXGI_FORMAT_R8G8_SNORM
DX_FORMAT_R8G8_SINT	52	DXGI_FORMAT_R8G8_SINT
DX_FORMAT_R16_TYPELESS	53	DXGI_FORMAT_R16_TYPELESS

© 2000-2025, MetaQuotes Ltd.

ID	값	Match in DXGI_FORMAT
DX_FORMAT_R16_FLOAT	54	DXGI_FORMAT_R16_FLOAT
DX_FORMAT_D16_UNORM	55	DXGI_FORMAT_D16_UNORM
DX_FORMAT_R16_UNORM	56	DXGI_FORMAT_R16_UNORM
DX_FORMAT_R16_UINT	57	DXGI_FORMAT_R16_UINT
DX_FORMAT_R16_SNORM	58	DXGI_FORMAT_R16_SNORM
DX_FORMAT_R16_SINT	59	DXGI_FORMAT_R16_SINT
DX_FORMAT_R8_TYPELESS	60	DXGI_FORMAT_R8_TYPELESS
DX_FORMAT_R8_UNORM	61	DXGI_FORMAT_R8_UNORM
DX_FORMAT_R8_UINT	62	DXGI_FORMAT_R8_UINT
DX_FORMAT_R8_SNORM	63	DXGI_FORMAT_R8_SNORM
DX_FORMAT_R8_SINT	64	DXGI_FORMAT_R8_SINT
DX_FORMAT_A8_UNORM	65	DXGI_FORMAT_A8_UNORM
DX_FORMAT_R1_UNORM	66	DXGI_FORMAT_R1_UNORM
DX_FORMAT_R9G9B9E5_SHAREDEXP	67	DXGI_FORMAT_R9G9B9E5_SHAREDEXP
DX_FORMAT_R8G8_B8G8_UNORM	68	DXGI_FORMAT_R8G8_B8G8_UNORM
DX_FORMAT_G8R8_G8B8_UNORM	69	DXGI_FORMAT_G8R8_G8B8_UNORM
DX_FORMAT_BC1_TYPELESS	70	DXGI_FORMAT_BC1_TYPELESS
DX_FORMAT_BC1_UNORM	71	DXGI_FORMAT_BC1_UNORM
DX_FORMAT_BC1_UNORM_SRGB	72	DXGI_FORMAT_BC1_UNORM_SRGB
DX_FORMAT_BC2_TYPELESS	73	DXGI_FORMAT_BC2_TYPELESS
DX_FORMAT_BC2_UNORM	74	DXGI_FORMAT_BC2_UNORM
DX_FORMAT_BC2_UNORM_SRGB	75	DXGI_FORMAT_BC2_UNORM_SRGB
DX_FORMAT_BC3_TYPELESS	76	DXGI_FORMAT_BC3_TYPELESS
DX_FORMAT_BC3_UNORM	77	DXGI_FORMAT_BC3_UNORM
DX_FORMAT_BC3_UNORM_SRGB	78	DXGI_FORMAT_BC3_UNORM_SRGB
DX_FORMAT_BC4_TYPELESS	79	DXGI_FORMAT_BC4_TYPELESS
DX_FORMAT_BC4_UNORM	80	DXGI_FORMAT_BC4_UNORM
DX_FORMAT_BC4_SNORM	81	DXGI_FORMAT_BC4_SNORM
DX_FORMAT_BC5_TYPELESS	82	DXGI_FORMAT_BC5_TYPELESS
DX_FORMAT_BC5_UNORM	83	DXGI_FORMAT_BC5_UNORM
DX_FORMAT_BC5_SNORM	84	DXGI_FORMAT_BC5_SNORM

© 2000-2025, MetaQuotes Ltd.

ID	값	Match in DXGI_FORMAT
DX_FORMAT_B5G6R5_UNORM	85	DXGI_FORMAT_B5G6R5_UNORM
DX_FORMAT_B5G5R5A1_UNORM	86	DXGI_FORMAT_B5G5R5A1_UNORM
DX_FORMAT_B8G8R8A8_UNORM	87	DXGI_FORMAT_B8G8R8A8_UNORM
DX_FORMAT_B8G8R8X8_UNORM	88	DXGI_FORMAT_B8G8R8X8_UNORM
DX_FORMAT_R10G10B10_XR_BIAS_A2_UNORM	89	DXGI_FORMAT_R10G10B10_XR_BIAS_A2_UNORM
DX_FORMAT_B8G8R8A8_TYPELESS	90	DXGI_FORMAT_B8G8R8A8_TYPELESS
DX_FORMAT_B8G8R8A8_UNORM_SRGB	91	DXGI_FORMAT_B8G8R8A8_UNORM_SRGB
DX_FORMAT_B8G8R8X8_TYPELESS	92	DXGI_FORMAT_B8G8R8X8_TYPELESS
DX_FORMAT_B8G8R8X8_UNORM_SRGB	93	DXGI_FORMAT_B8G8R8X8_UNORM_SRGB
DX_FORMAT_BC6H_TYPELESS	94	DXGI_FORMAT_BC6H_TYPELESS
DX_FORMAT_BC6H_UF16	95	DXGI_FORMAT_BC6H_UF16
DX_FORMAT_BC6H_SF16	96	DXGI_FORMAT_BC6H_SF16
DX_FORMAT_BC7_TYPELESS	97	DXGI_FORMAT_BC7_TYPELESS
DX_FORMAT_BC7_UNORM	98	DXGI_FORMAT_BC7_UNORM
DX_FORMAT_BC7_UNORM_SRGB	99	DXGI_FORMAT_BC7_UNORM_SRGB
DX_FORMAT_AYUV	100	DXGI_FORMAT_AYUV
DX_FORMAT_Y410	101	DXGI_FORMAT_Y410
DX_FORMAT_Y416	102	DXGI_FORMAT_Y416
DX_FORMAT_NV12	103	DXGI_FORMAT_NV12
DX_FORMAT_P010	104	DXGI_FORMAT_P010
DX_FORMAT_P016	105	DXGI_FORMAT_P016
DX_FORMAT_420_OPAQUE	106	DXGI_FORMAT_420_OPAQUE
DX_FORMAT_YUY2	107	DXGI_FORMAT_YUY2
DX_FORMAT_Y210	108	DXGI_FORMAT_Y210
DX_FORMAT_Y216	109	DXGI_FORMAT_Y216
DX_FORMAT_NV11	110	DXGI_FORMAT_NV11
DX_FORMAT_AI44	111	DXGI_FORMAT_AI44
DX_FORMAT_IA44	112	DXGI_FORMAT_IA44
DX_FORMAT_P8	113	DXGI_FORMAT_P8

© 2000-2025, MetaQuotes Ltd.

ID	값	Match in DXGI_FORMAT
DX_FORMAT_A8P8	114	DXGI_FORMAT_A8P8
DX_FORMAT_B4G4R4A4_UNORM	115	DXGI_FORMAT_B4G4R4A4_UNORM
DX_FORMAT_P208	130	DXGI_FORMAT_P208
DX_FORMAT_V208	131	DXGI_FORMAT_V208
DX_FORMAT_V408	132	DXGI_FORMAT_V408
DX_FORMAT_FORCE_UINT	0xffffffff	DXGI_FORMAT_FORCE_UINT

© 2000-2025, MetaQuotes Ltd.

DXInputCreate

셰이더 입력 생성.

```
int  DXInputCreate(
   int   context,        // 그래픽 컨텍스트 핸들
   uint  input_size      // 입력 크기(바이트)
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

input_size

[in] 매개변수 구조의 크기(바이트).

값 반환

오류 발생 시 셰이더 입력용 핸들 또는 INVALID_HANDLE. 오류 코드를 받으려면 GetLastError() 기능을 호출해야 합니다.

더 이상 사용되지 않는 생성된 핸들은 DXRelease() 기능에 의해 명시적으로 해제되어야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXInputSet

셰이더 입력 설정.

```
bool  DXInputSet(
   int           input,      // 그래픽 컨텍스트 핸들
   const void&   data        // 설정할 데이터
   );
```

매개변수

입력

[in] DXInputCreate()에서 얻은 셰이더 입력 핸들.

데이터

[in] 셰이더 입력 설정 데이터.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXShaderCreate

지정된 유형의 셰이더를 생성.

```
int  DXShaderCreate(
   int                  context,          // 그래픽 컨텍스트 핸들
   ENUM_DX_SHADER_TYPE  shader_type,      // 셰이더 유형
   const string         source,           // 셰이더 소스 코드
   const string         entry_point,      // 진입 포인트
   string&              compile_error     // 컴파일러 메시지를 수신하는 문자열
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

shader_type

[out] ENUM_DX_SHADER_TYPE 열거 값.

소스

[in] HLSL 5의 셰이더 소스 코드.

entry_point

[in] 진입 포인트 – 소스 코드의 함수 이름.

compile_error

[in] 컴파일 오류를 수신하기 위한 문자열.

값 반환

에러 발생 시 셰이더 또는 INVALID_HANDLE을 처리. 오류 코드를 받으려면 GetLastError() 기능을 호출해야 합니다.

주의

더 이상 사용되지 않는 생성된 핸들은 DXRelease() 기능에 의해 명시적으로 해제되어야 합니다.

ENUM_DX_SHADER_TYPE

ID	값	설명
DX_SHADER_VERTEX	0	정점 셰이더
DX_SHADER_GEOMETRY	1	지오메트리 셰이더
DX_SHADER_PIXEL	2	픽셀 셰이더

© 2000-2025, MetaQuotes Ltd.

DXShaderSetLayout

정점 셰이더의 정점 레이아웃을 설정.

```
bool  DXShaderSetLayout(
   int                     shader,       // 셰이더 핸들
   const DXVertexLayout&   layout[]      //
   );
```

매개변수

셰이더

[in] DXShaderCreate()에서 생성된 정점 셰이더의 핸들.

layout[]

[in] 정점 필드 설명의 배열. 설명은 DXVertexLayout 구조로 설정됩니다:

```
struct DXVertexLayout
  {
   string          semantic_name;       // 셰이더 입력 서명에서 이 요소와 관련된 HLSL 의
   uint            semantic_index;      // 요소의 의미 인덱스. 의미 인덱스는 정수 인덱스
   ENUM_DX_FORMAT  format;              // 요소 데이터의 데이터 유형.
  };
```

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

레이아웃은 지정된 정점 버퍼의 정점 유형과 일치해야 합니다. 또한 정점 셰이더 코드의 진입 포인트에서 사용되는 정점의 입력 유형과 일치해야 합니다.

셰이더의 정점 버퍼는 DXBufferSet()에 설정되어 있습니다.

DXVertexLayout 구조는 D3D11_INPUT_ELEMENT_DESC MSDN 구조의 버전입니다.

© 2000-2025, MetaQuotes Ltd.

DXShaderInputsSet

셰이더 입력 설정.

```
bool  DXShaderInputsSet(
   int          shader,        // 셰이더 핸들
   const int&   inputs[]       // 입력 핸들 배열
   );
```

매개변수

셰이더

[in] DXShaderCreate()에 생성된 셰이더의 핸들.

inputs[]

[in] DXInputCreate()를 사용하여 생성된 입력 핸들 배열.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

입력 매개변수의 크기는 셰이더 코드에 선언된 cbuffer 개채 수와 같아야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXShaderTexturesSet

셰이더 텍스쳐를 설정.

```
bool  DXShaderTexturesSet(
   int           shader,          // 셰이더 핸들
   const  int&   textures[]       // 구조 핸들의 배열
   );
```

매개변수

셰이더

[in] DXShaderCreate()에 생성된 셰이더의 핸들.

textures[]

[in] DXTextureCreate()를 사용하여 생성된 텍스처 핸들 배열.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

텍스처 배열의 크기는 셰이더 코드에 언급된 Texture2D 개체 수와 같아야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXDraw

DXBufferSet()에 있는 정점 버퍼 세트의 정점을 렌더링.

```
bool  DXDraw(
   int   context,              // 그래픽 컨텍스트 핸들
   uint  start=0,              // 첫 번째 정점 인덱스
   uint  count=WHOLE_ARRAY     // 꼭지점 수
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

start=0

[in] 렌더링을 위한 첫 번째 정점의 인덱스.

count=WHOLE_ARRAY

[in] 렌더링할 정점의 수.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

정점을 렌더링하기 위해 DXShaderSet()를 사용하여 셰이더를 미리 설정해야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXDrawIndexed

DXBufferSet()에서 인덱스 버퍼가 설명하는 그래픽 원형을 렌더링.

```
bool  DXDrawIndexed(
   int   context,               // 그래픽 컨텍스트 핸들
   uint  start=0,               // 첫 번째 원형 인덱스
   uint  count=WHOLE_ARRAY      // 원형 숫자
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

start=0

[in] 렌더링을 위한 첫 번째 원형 인덱스.

count=WHOLE_ARRAY

[in] 렌더링을 위한 기초 수

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

인덱스 버퍼에서 설명하는 원형 유형은 DXPrimiveTopologySet()를 사용하여 설정됩니다.

DXBufferSet()의 정점 버퍼는 기본적으로 원형을 렌더링하도록 설정해야 합니다.

또한 셰이더는 DXShaderSet()를 사용하여 예비적으로 설정해야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXPrimiveTopologySet

DXDrawIndexed()를 사용하여 렌더링할 기본 요소 유형을 설정.

```
bool  DXPrimiveTopologySet(
   int                         context,              // 그래픽 컨텍스트 핸들
   ENUM_DX_PRIMITIVE_TOPOLOGY  primitive_topology    // 원형
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

primitive_topology

[in] ENUM_DX_PRIMITIVE_TOPOLOGY 열거 값enumeration.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

ENUM_DX_PRIMITIVE_TOPOLOGY

ID	값	부합:D3D11_PRIMITIVE_TOPOLOGY
DX_PRIMITIVE_TOPOLOGY_POINTLIST	1	D3D11_PRIMITIVE_TOPOLOGY_POINTLIST
DX_PRIMITIVE_TOPOLOGY_LINELIST	2	D3D11_PRIMITIVE_TOPOLOGY_LINELIST
DX_PRIMITIVE_TOPOLOGY_LINESTRIP	3	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST	4	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP	5	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP
DX_PRIMITIVE_TOPOLOGY_LINELIST_ADJ	6	D3D11_PRIMITIVE_TOPOLOGY_LINELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ	7	D3D11_PRIMITIVE_TOPOLOGY_LINESTRIP_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ	8	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLELIST_ADJ
DX_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ	9	D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP_ADJ

© 2000-2025, MetaQuotes Ltd.

DXBufferSet

현재 렌더링에 대한 버퍼를 설정

```
bool  DXBufferSet(
   int   context,              // 그래픽 컨텍스트 핸들
   int   buffer,               // 정점 또는 인덱스 버퍼 핸들
   uint  start=0,              // 초기 인덱스
   uint  count=WHOLE_ARRAY     // 요소의 수
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

buffer

[in] DXBufferCreate()에서 생성된 정점 또는 인덱스 버퍼 핸들.

start=0

[in] 버퍼 첫 번째 요소의 인덱스. 버퍼 시작 부분의 데이터가 기본적으로 사용됩니다.

count=WHOLE_ARRAY

[in] 사용할 값의 수. 기본값은 모든 버퍼의 값입니다.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

The DXBufferSet() 를 호출하는데 렌더링을 하기 위한 정점 및 인덱스 버퍼를 설정하는 데에는 DXDraw()를 사용해야 합니다.

© 2000-2025, MetaQuotes Ltd.

DXShaderSet

렌더링을 위한 셰이더 설정.

```
bool  DXShaderSet(
   int  context,      // 그래픽 컨텍스트 핸들
   int  shader        // 셰이더 핸들
   );
```

매개변수

컨텍스트

[in] DXContextCreate()에서 작성된 그래픽 컨텍스트를 처리.

셰이더

[in] DXShaderCreate()에 생성된 셰이더의 핸들.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

렌더링에 여러 유형의 셰이더를 동시에 사용할 수 있습니다(정점, 지오메트리 및 픽셀1).

© 2000-2025, MetaQuotes Ltd.

DXHandleType

Returns a handle type.

```
ENUM_DX_HANDLE_TYPE  DXHandleType(
   int  handle      // 핸들
   );
```

매개변수

핸들

[in] 핸들.

값 반환

ENUM_DX_HANDLE_TYPE 열거 값

ENUM_DX_HANDLE_TYPE

ID	값	설명
DX_HANDLE_INVALID	0	유효하지 않은 핸들
DX_HANDLE_CONTEXT	1	그래픽 컨텍스트 핸들
DX_HANDLE_SHADER	2	셰이더 핸들
DX_HANDLE_BUFFER	3	정점 또는 인덱스 버퍼 핸들
DX_HANDLE_INPUT	4	셰이더 입력 핸들
DX_HANDLE_TEXTURE	5	텍스쳐 핸들

© 2000-2025, MetaQuotes Ltd.

DXRelease

Releases a handle.

```
bool  DXRelease(
   int  handle      // 핸들
   );
```

매개변수

컨텍스트

[in] 해제된 핸들.

값 반환

실행에 성공하면 true를, 그렇지 않으면 false를 반환합니다. 오류 코드를 수신하려면 GetLastError() 함수를 호출해야 합니다.

주의

더 이상 사용되지 않는 생성된 모든 핸들은 DXRelease() 기능을 통해 명시적으로 해제해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Python과의 통합을 위한 MetaTrader 모듈

MQL5는 금융 시장에서 고성능 트레이딩 애플리케이션을 개발하기 위해 고안되었으며, 알고리즘 트레이딩에 사용되는 다른 전문 언어 중에서 비교할 수 없습니다. MQL5 프로그램의 구문과 속도는 가능한 한 C++에 가까우며, OpenCL 및 MS 비쥬얼 스튜디오와의 통합이 지원됩니다. 통계, 퍼지 로직 및 ALGLIB 라이브러리도 사용할 수 있습니다. MetaEditor 개발 환경은 "스마트" 기능 가져오기 기능을 통해 기본적으로 .NET 라이브러리 지원하므로 특별한 래퍼 개발이 필요하지 않습니다. 타사 C++ DLL도 사용할 수 있습니다. C++ 소스 코드 파일(CPP 및 H)은 편집기에서 직접 편집하여 DLL로 컴파일할 수 있습니다. 사용자의 PC에 설치된 마이크로소프트 비쥬얼 스튜디오를 이를 위해 사용할 수 있습니다.

Python은 스크립트 및 애플리케이션 개발을 위한 현대적인 고급 프로그래밍 언어입니다. 여기에는 머신러닝, 프로세스 자동화, 데이터 분석 및 시각화를 위한 여러 라이브러리가 포함되어 있습니다.

Python용 MetaTrader 패키지는 MetaTrader 5 터미널에서 프로세서 간 통신을 통해 교환 데이터를 쉽고 빠르게 얻을 수 있도록 설계되었습니다. 이러한 방식으로 수신된 데이터는 통계 계산 및 기계 학습에 추가로 사용될 수 있습니다.

명령줄에서 패키지 설치:

```
pip install MetaTrader5
```

명령줄에서 패키지 업데이트:

```
pip install --upgrade MetaTrader5
```

MetaTrader 5 및 Python 통합 기능

Function	Action
initialize	MetaTrader 5 터미널과의 연결 설정
login	MetaTrader 5 터미널과의 연결 설정
shutdown	이전에 설정한 MetaTrader 5 터미널 연결을 닫습니다
version	MetaTrader 5 터미널 버전 반환
last_error	마지막 오류에 대한 데이터 반환
account_info	현재 거래 계정에 대한 정보를 가져옵니다
terminal_Info	연결된 MetaTrader 5 터미널의 상태 및 매개 변수를 가져옵니다
symbols_total	MetaTrader 5 터미널에 있는 모든 금융상품의 수
symbols_get	MetaTrader 5 터미널에서 모든 금융상품을 가져옵니다
symbol_info	지정된 금융상품에 대한 데이터을 가져옵니다
symbol_info_tick	지정된 금융상품에 대한 마지막 틱을 가져옵니다
symbol_select	MarketWatch 창에서 기호를 선택하거나 창에서 기호를 제거합니다
market_book_add	MetTrader 5 터미널을 특정 심볼에 대한 Market Depth 변경 이벤트에 등록합니다.
market_book_get	지정된 기호에 대한 Market Depth 항목이 포함된 BookInfo 튜플을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Function	Action
market_book_release	지정된 심볼에 대한 MetaTrader 5 터미널 Market Depth 변경 이벤트 구독 취소
copy_rates_from	지정된 날짜부터 MetaTrader 5 터미널에서 막대를 가져옵니다
copy_rates_from_pos	지정된 인덱스에서 시작하는 MetaTrader 5 터미널에서 막대를 가져옵니다
copyrates_range	MetaTrader 5 터미널에서 지정된 날짜 범위의 막대를 가져옵니다
copy_ticks_from	지정된 날짜부터 MetaTrader 5 터미널에서 틱을 가져옵니다
copy_ticks_range	MetaTrader 5 터미널에서 지정된 날짜 범위에 대한 틱을 가져옵니다
orders_total	활성 주문 수를 가져옵니다
orders_get	심볼 또는 티켓별로 필터링할 수 있는 기능으로 활성 주문을 가져옵니다
order_calc_margin	지정된 거래 작업을 수행할 계정 통화의 마진 반환
order_calc_profit	지정된 거래 작업에 대한 계정 통화로 수익 반환
order_check	다음에 대한 계정 통화로 수익 반환: 지정된 거래 작업
order_send	거래 작업을 수행하기 위한 요청 전송
positions_total	오픈 포지션 수를 가져옵니다
positions_get	심볼 또는 티켓별로 필터링할 수 있는 오픈 포지션을 가져옵니다
history_orders_total	지정된 간격 내에 거래 기록의 주문 수를 가져옵니다
history_orders_get	티켓 또는 포지션별로 필터링할 수 있는 기능으로 거래 내역에서 주문을 가져옵니다
history_deals_total	지정된 간격 내에 거래 기록의 거래 수를 가져옵니다
history_deals_get	티켓 또는 직급별로 필터링할 수 있는 기능으로 거래 기록에서 거래를 가져옵니다

Python을 MetaTrader 5에 연결하는 예

1. Python 3.8 최신 버전 다운로드 하기 : https://www.python.org/downloads/windows
2. Python을 설치할 때 명령줄에서 Python 스크립트를 실행할 수 있도록 하려면 "Add Python 3.8 to PATH %"를 체크하십시오.
3. 명령줄에서 MetaTrader 5 모듈 설치

```
pip install MetaTrader5
```

4. matplotlib 및 pandas 패키지 추가

```
pip install matplotlib
pip install pandas
```

5. 테스트 스크립트 실행

```
from datetime import datetime
import matplotlib.pyplot as plt
```

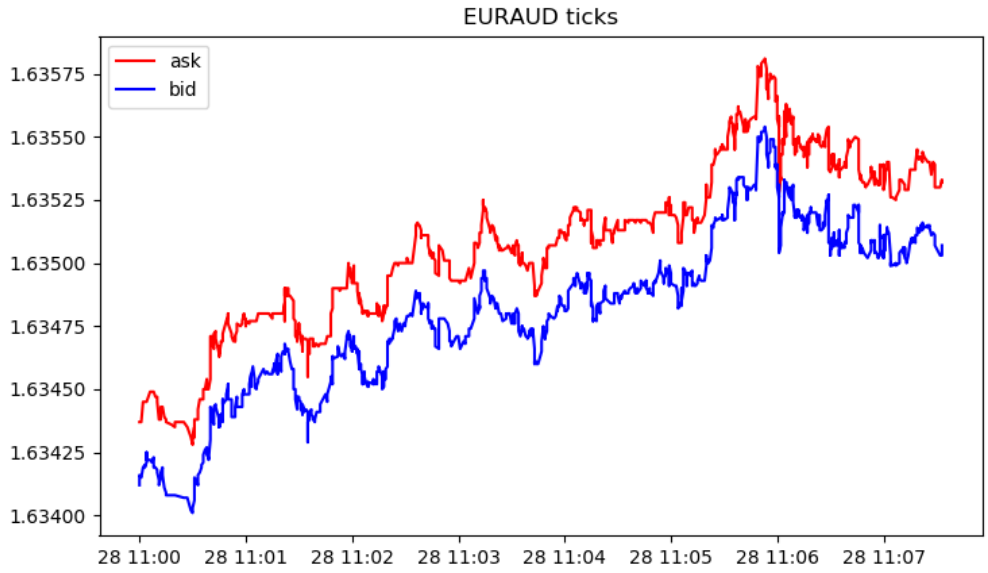
© 2000-2025, MetaQuotes Ltd.

```
import pandas as pd
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import MetaTrader5 as mt5

# MetaTrader 5 연결
if not mt5.initialize():
    print("initialize() 실패")
    mt5.shutdown()

# 연결 상태 및 매개 변수 요청
print(mt5.terminal_info())
# MetaTrader 5 버전에 대한 데이터를 가져옵니다
print(mt5.version())

# EURAUD에 1000 틱 요청
euraud_ticks = mt5.copy_ticks_from("EURAUD", datetime(2020,1,28,13), 1000, mt5.COPY_TI
# 2019.04.01 13:00 - 2019.04.02 13:00 이내에 AUDUSD로 틱 요청
audusd_ticks = mt5.copy_ticks_range("AUDUSD", datetime(2020,1,27,13), datetime(2020,1,

# 여러 가지 방법으로 다른 심볼에서 막대를 가져옵니다
eurusd_rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_M1, datetime(2020,1,28,13),
eurgbp_rates = mt5.copy_rates_from_pos("EURGBP", mt5.TIMEFRAME_M1, 0, 1000)
eurcad_rates = mt5.copy_rates_range("EURCAD", mt5.TIMEFRAME_M1, datetime(2020,1,27,13)

# MetaTrader 5에 대한 연결 종료
mt5.shutdown()

#DATA
print('euraud_ticks(', len(euraud_ticks), ')')
for val in euraud_ticks[:10]: print(val)

print('audusd_ticks(', len(audusd_ticks), ')')
for val in audusd_ticks[:10]: print(val)

print('eurusd_rates(', len(eurusd_rates), ')')
for val in eurusd_rates[:10]: print(val)

print('eurgbp_rates(', len(eurgbp_rates), ')')
for val in eurgbp_rates[:10]: print(val)

print('eurcad_rates(', len(eurcad_rates), ')')
for val in eurcad_rates[:10]: print(val)

#PLOT
# 가져온 데이터로 DataFrame 생성
ticks_frame = pd.DataFrame(euraud_ticks)
# 시간(초)을 날짜 시간 형식으로 변환
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')
```

© 2000-2025, MetaQuotes Ltd.

```
# 차트에 틱 표시
plt.plot(ticks_frame['time'], ticks_frame['ask'], 'r-', label='ask')
plt.plot(ticks_frame['time'], ticks_frame['bid'], 'b-', label='bid')

# 레전드 표시
plt.legend(loc='upper left')

# 헤더 추가
plt.title('EURAUD ticks')

# 차트 표시
plt.show()
```

6. 데이터 및 차트를 가져옵니다




```
[2, 'MetaQuotes-Demo', '16167573']
[500, 2325, '19 Feb 2020']

euraud_ticks( 1000 )
(1580209200, 1.63412, 1.63437, 0., 0, 1580209200067, 130, 0.)
(1580209200, 1.63416, 1.63437, 0., 0, 1580209200785, 130, 0.)
(1580209201, 1.63415, 1.63437, 0., 0, 1580209201980, 130, 0.)
(1580209202, 1.63419, 1.63445, 0., 0, 1580209202192, 134, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203004, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203487, 130, 0.)
(1580209203, 1.6342, 1.63445, 0., 0, 1580209203694, 130, 0.)
(1580209203, 1.63419, 1.63445, 0., 0, 1580209203990, 130, 0.)
(1580209204, 1.63421, 1.63445, 0., 0, 1580209204194, 130, 0.)
(1580209204, 1.63425, 1.63445, 0., 0, 1580209204392, 130, 0.)
audusd_ticks( 40449 )
(1580122800, 0.67858, 0.67868, 0., 0, 1580122800244, 130, 0.)
(1580122800, 0.67858, 0.67867, 0., 0, 1580122800429, 4, 0.)
(1580122800, 0.67858, 0.67865, 0., 0, 1580122800817, 4, 0.)
```

© 2000-2025, MetaQuotes Ltd.

```
(1580122801, 0.67858, 0.67866, 0., 0, 1580122801618, 4, 0.)
(1580122802, 0.67858, 0.67865, 0., 0, 1580122802928, 4, 0.)
(1580122809, 0.67855, 0.67865, 0., 0, 1580122809526, 130, 0.)
(1580122809, 0.67855, 0.67864, 0., 0, 1580122809699, 4, 0.)
(1580122813, 0.67855, 0.67863, 0., 0, 1580122813576, 4, 0.)
(1580122815, 0.67856, 0.67863, 0., 0, 1580122815190, 130, 0.)
(1580122815, 0.67855, 0.67863, 0., 0, 1580122815479, 130, 0.)
eurusd_rates( 1000 )
(1580149260, 1.10132, 1.10151, 1.10131, 1.10149, 44, 1, 0)
(1580149320, 1.10149, 1.10161, 1.10143, 1.10154, 42, 1, 0)
(1580149380, 1.10154, 1.10176, 1.10154, 1.10174, 40, 2, 0)
(1580149440, 1.10174, 1.10189, 1.10168, 1.10187, 47, 1, 0)
(1580149500, 1.10185, 1.10191, 1.1018, 1.10182, 53, 1, 0)
(1580149560, 1.10182, 1.10184, 1.10176, 1.10183, 25, 3, 0)
(1580149620, 1.10183, 1.10187, 1.10177, 1.10187, 49, 2, 0)
(1580149680, 1.10187, 1.1019, 1.1018, 1.10187, 53, 1, 0)
(1580149740, 1.10187, 1.10202, 1.10187, 1.10198, 28, 2, 0)
(1580149800, 1.10198, 1.10198, 1.10183, 1.10188, 39, 2, 0)
eurgbp_rates( 1000 )
(1582236360, 0.83767, 0.83767, 0.83764, 0.83765, 23, 9, 0)
(1582236420, 0.83765, 0.83765, 0.83764, 0.83765, 15, 8, 0)
(1582236480, 0.83765, 0.83766, 0.83762, 0.83765, 19, 7, 0)
(1582236540, 0.83765, 0.83768, 0.83758, 0.83763, 39, 6, 0)
(1582236600, 0.83763, 0.83768, 0.83763, 0.83767, 21, 6, 0)
(1582236660, 0.83767, 0.83775, 0.83765, 0.83769, 63, 5, 0)
(1582236720, 0.83769, 0.8377, 0.83758, 0.83764, 40, 7, 0)
(1582236780, 0.83766, 0.83769, 0.8376, 0.83766, 37, 6, 0)
(1582236840, 0.83766, 0.83772, 0.83763, 0.83772, 22, 6, 0)
(1582236900, 0.83772, 0.83773, 0.83768, 0.8377, 36, 5, 0)
eurcad_rates( 1441 )
(1580122800, 1.45321, 1.45329, 1.4526, 1.4528, 146, 15, 0)
(1580122860, 1.4528, 1.45315, 1.45274, 1.45301, 93, 15, 0)
(1580122920, 1.453, 1.45304, 1.45264, 1.45264, 82, 15, 0)
(1580122980, 1.45263, 1.45279, 1.45231, 1.45277, 109, 15, 0)
(1580123040, 1.45275, 1.4528, 1.45259, 1.45271, 53, 14, 0)
(1580123100, 1.45273, 1.45285, 1.45269, 1.4528, 62, 16, 0)
(1580123160, 1.4528, 1.45284, 1.45267, 1.45282, 64, 14, 0)
(1580123220, 1.45282, 1.45299, 1.45261, 1.45272, 48, 14, 0)
(1580123280, 1.45272, 1.45275, 1.45255, 1.45275, 74, 14, 0)
(1580123340, 1.45275, 1.4528, 1.4526, 1.4528, 94, 13, 0)
```

© 2000-2025, MetaQuotes Ltd.

initialize

MetaTrader 5 단자와 연결을 설정합니다. 호출 옵션은 세 가지가 있습니다.

매개 변수 없이 호출. 연결을 위한 터미널이 자동으로 검색됨.

```
initialize()
```

연결하고자 하는 MetaTrader 5 터미널의 경로를 지정하는 호출.

```
initialize(
   path                       // MetaTrader 5 터미널 EXE 파일의 경로
   )
```

거래 계정 경로 및 매개 변수를 지정하는 호출.

```
initialize(
   path,                      // MetaTrader 5 터미널 EXE 파일의 경로
   login=LOGIN,               // 계정 번호
   password="PASSWORD",       // 비밀번호
   server="SERVER",           // 터미널에 지정된 서버 이름
   timeout=TIMEOUT,           // timeout
   portable=False             // portable mode
   )
```

Parameters

path

[in] metatrader.exe 또는 metatrader64.exe 파일로의 경로. 이름이 지정되지 않은 매개 변수(선택 사항). 먼저 파라미터 이름 없이 표시됩니다. 경로를 지정하지 않으면 모듈이 자체적으로 실행 파일을 찾으려고 시도합니다.

login=LOGIN

[in] 거래 계정 번호. 선택적 지명 파라미터. 지정하지 않으면 마지막 거래 계정이 사용됩니다.

password="PASSWORD"

[in] 거래 계정 비밀번호 선택적 지명 파라미터. 비밀번호가 설정되어 있지 않으면 터미널 데이터베이스에 저장된 특정 거래 계정의 비밀번호가 자동으로 적용됩니다.

server="SERVER"

[in] 거래 서버명 선택적 지명 파라미터. 서버가 설정되지 않은 경우 터미널 데이터베이스에 저장된 특정 거래 계정의 서버가 자동으로 적용됩니다.

timeout=TIMEOUT

[in] 연결 시간 제한(밀리초). 선택적 지명 파라미터. 지정하지 않으면 값이 60000(60초)로 적용됩니다.

portable=False

[in] 휴대용 모드에서의 터미널 시작 플래그. 선택적 지명 파라미터. 지정하지 않으면 False 값이 사용됩니다.

반환 값

MetaTrader 5 터미널에 성공적으로 연결되면 True를 반환하고, 그렇지 않으면 False를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

참고

필요한 경우, initialize() 호출을 실행할 때 연결을 설정하기 위해 MetaTrader 5 터미널을 시작합니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# 지정된 거래 계정에 대한 MetaTrader 5 연결 설정
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 연결 상태, 서버 이름 및 거래 계정의 데이터 표시
print(mt5.terminal_info())
# MetaTrader 5 버전에서 데이터 표시
print(mt5.version())

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

더 보기

shutdown, terminal_info, version

© 2000-2025, MetaQuotes Ltd.

login

지정된 매개 변수를 사용하여 거래 계정에 연결합니다.

```
login(
   login,                    // 계정 번호
   password="PASSWORD",      // password
   server="SERVER",          // 터미널에 지정된 서버 이름
   timeout=TIMEOUT           // timeout
   )
```

Parameters

login

[in] 거래 계정 번호. 이름이 지정되지 않은 필수 매개 변수.

password

[in] 거래 계정 비밀번호 선택적 지명 파라미터. 비밀번호가 설정되어 있지 않으면 터미널 데이터베이스에 저장된 비밀번호가 자동으로 적용됩니다.

server

[in] 거래 서버명 선택적 지명 파라미터. 서버가 설정되지 않은 경우 마지막으로 사용한 서버가 자동으로 적용됩니다.

timeout=TIMEOUT

[in] 연결 시간 제한(밀리초). 선택적 지명 파라미터. 지정하지 않으면 값이 60000(60초)로 적용됩니다. 지정된 시간 내에 연결이 설정되지 않으면 통화가 강제로 종료되고 예외가 생성됩니다.

반환 값

거래 계정에 성공적으로 연결된 경우 True이며, 그렇지 않은 경우 False입니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# MetaTrader 5 버전에서 데이터 표시
print(mt5.version())
# 비밀번호와 서버를 지정하지 않고 거래 계정에 연결합니다.
account=17221085
authorized=mt5.login(account)  # 터미널 데이터베이스 비밀번호가 연결 데이터를 기억하도록 설정
if authorized:
    print("다음 계정에 연결 #{}".format(account))
else:
```

© 2000-2025, MetaQuotes Ltd.

```
    print("계정 연결 실패 #{}, 오류 코드: {}".format(account, mt5.last_error()))

# 이제 비밀번호를 지정하는 다른 거래 계정에 연결합니다.
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
    # 거래 계정 데이터를 "있는 그대로" 표시
    print(mt5.account_info())
    # 거래 계정 데이터를 목록 형식으로 표시
    print("Show account_info()._asdict():")
    account_info_dict = mt5.account_info()._asdict()
    for prop in account_info_dict:
        print("  {}={}".format(prop, account_info_dict[prop]))
else:
    print("계정 연결 실패 #{}, 오류 코드: {}".format(account, mt5.last_error()))

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()


Result:
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29
[500, 2367, '23 Mar 2020']

계정 #17221085에 연결

계정 #25115284에 연결
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
계정 속성:
   login=25115284
   trade_mode=0
   leverage=100
   limit_orders=200
   margin_so_mode=0
   trade_allowed=True
   trade_expert=True
   margin_mode=2
   currency_digits=2
   fifo_close=False
   balance=99588.33
   credit=0.0
   profit=-45.23
   equity=99543.1
   margin=54.37
   margin_free=99488.73
   margin_level=183084.6054809638
   margin_so_call=50.0
   margin_so_so=30.0
```

© 2000-2025, MetaQuotes Ltd.

```
    margin_initial=0.0
    margin_maintenance=0.0
    assets=0.0
    liabilities=0.0
    commission_blocked=0.0
    name=James Smith
    server=MetaQuotes-Demo
    currency=USD
    company=MetaQuotes Software Corp.
```

더 보기

initialize, shutdown

© 2000-2025, MetaQuotes Ltd.

shutdown

이전에 설정한 MetaTrader 5 터미널 연결을 종료합니다.

```
shutdown()
```

반환 값

None.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패")
    quit()

# 연결 상태, 서버 이름 및 거래 계정의 데이터 표시
print(mt5.terminal_info())
# MetaTrader 5 버전에서 데이터 표시
print(mt5.version())

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

더 보기

initialize, login_py, terminal_info, version

© 2000-2025, MetaQuotes Ltd.

version

MetaTrader 5 터미널 버전을 반환합니다.

```
version()
```

반환 값

MetaTrader 5 터미널 버전, 빌드 및 릴리즈 날짜를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

version() 함수는 터미널 버전, 빌드 및 릴리즈 날짜를 다음 세 값의 튜플로서 반환합니다.

Type	Description	샘플 값
integer	MetaTrader 5 터미널 버전	500
integer	Build	2007
string	Build release date	'25 Feb 2019'

예:

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# MetaTrader 5 버전에서 데이터 표시
print(mt5.version())

# 연결 상태, 서버 이름 및 거래 계정의 데이터를 '있는 그대로' 표시
print(mt5.terminal_info())
print()

# 딕셔너리 형태로 속성 가져오기
terminal_info_dict=mt5.terminal_info()._asdict()
# 딕셔너리를 데이터프레임으로 변환하여 출력
df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
print("terminal_info() as dataframe:")
print(df[:-1])

# MetaTrader 5 터미널 연결 종료
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()


Result:
MetaTrader 5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader 5 패키지 버전:  5.0.29
[500, 2367, '23 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True, dlls_a

terminal_info() as dataframe:
                속성                          값
0       community_account                       True
1    community_connection                       True
2               connected                       True
3            dlls_allowed                      False
4           trade_allowed                      False
5       tradeapi_disabled                      False
6           email_enabled                      False
7             ftp_enabled                      False
8   notifications_enabled                      False
9                    mqid                      False
10                  build                       2367
11                maxbars                       5000
12               codepage                       1251
13             ping_last                      77881
14      community_balance                    707.107
15         retransmission                          0
16                company     MetaQuotes Software Corp.
17                   name                MetaTrader 5
18               language                    Russian
19                   path  E:\ProgramFiles\MetaTrader 5
20              data_path  E:\ProgramFiles\MetaTrader 5
```

더 보기

initialize, shutdown, terminal_info

© 2000-2025, MetaQuotes Ltd.

last_error

마지막 오류에 대한 데이터를 반환합니다.

```
last_error()
```

반환 값

마지막 오류 코드와 설명을 튜플로 반환합니다.

참고

last_error()를 사용하면 MetaTrader 5 라이브러리 기능의 실행에 실패한 경우 오류 코드를 얻을 수 있습니다. 이것은 GetLastError()와 유사합니다. 그러나 자체 오류 코드가 적용됩니다. 가능한 값:

Constant	Value	Description
RES_S_OK	1	대체로 성공
RES_E_FAIL	-1	대체로 실패
RES_E_INVALID_PARAMS	-2	잘못된 인수/매개 변수
RES_E_NO_MEMORY	-3	no memory condition
RES_E_NOT_FOUND	-4	내역 없음
RES_E_INVALID_VERSION	-5	잘못된 버전
RES_E_AUTH_FAILED	-6	인증 실패
RES_E_UNSUPPORTED	-7	지원되지 않는 방법
RES_E_AUTO_TRADING_DISABLED	-8	자동거래 비활성화
RES_E_INTERNAL_FAIL	-10000	내부 IPC 일반 오류
RES_E_INTERNAL_FAIL_SEND	-10001	내부 IPC 전송 실패
RES_E_INTERNAL_FAIL_RECEIVE	-10002	내부 IPC recv 실패
RES_E_INTERNAL_FAIL_INIT	-10003	내부 IPC 초기화 실패
RES_E_INTERNAL_FAIL_CONNECT	-10003	내부 IPC no ipc
RES_E_INTERNAL_FAIL_TIMEOUT	-10005	내부 시간초과

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()
```

© 2000-2025, MetaQuotes Ltd.

```
# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

더 보기

version, GetLastError

© 2000-2025, MetaQuotes Ltd.

account_info

현재 거래 계정에 대한 정보를 가져옵니다.

```
account_info()
```

반환 값

명명된 튜플 구조(이름 지정된 튜플)의 형식으로 정보를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

이 함수는 한번의 호출에서 AccountInfoInteger, AccountInfoDouble 및 AccountInfoString을 사용하여 얻을 수 있는 모든 데이터를 반환합니다.

예:

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 비밀번호 및 서버를 입력하는 거래 계정에 연결
authorized=mt5.login(25115284, password="gqz0343lbdm")
if authorized:
    account_info=mt5.account_info()
    if account_info!=None:
        # 거래 계정 데이터를 '있는 그대로' 표시
        print(account_info)
        # 거래 계정 자료를 딕셔너리 형태로 표시
        print("Show account_info()._asdict():")
        account_info_dict = mt5.account_info()._asdict()
        for prop in account_info_dict:
            print("  {}={}".format(prop, account_info_dict[prop]))
        print()

       # 딕셔너리를 데이터프레임으로 변환하여 출력
        df=pd.DataFrame(list(account_info_dict.items()),columns=['property','value'])
        print("account_info() as dataframe:")
        print(df)
else:
    print("비밀번호=gqz0343l bdm을 사용하여 거래 계정 25115284에 연결하지 못했음, 오류 코드 ="

# MetaTrader 5 터미널 연결 종료
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()


결과:
MetaTrader 5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader 5 패키지 버전:  5.0.29
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
Show account_info()._asdict():
  login=25115284
  trade_mode=0
  leverage=100
  limit_orders=200
  margin_so_mode=0
  trade_allowed=True
  trade_expert=True
  margin_mode=2
  currency_digits=2
  fifo_close=False
  balance=99511.4
  credit=0.0
  profit=41.82
  equity=99553.22
  margin=98.18
  margin_free=99455.04
  margin_level=101398.67590140559
  margin_so_call=50.0
  margin_so_so=30.0
  margin_initial=0.0
  margin_maintenance=0.0
  assets=0.0
  liabilities=0.0
  commission_blocked=0.0
  server=MetaQuotes-Demo
  currency=USD
  company=MetaQuotes Software Corp.

account_info() as dataframe
             속성                        값
0                login                 25115284
1           trade_mode                        0
2             leverage                      100
3         limit_orders                      200
4       margin_so_mode                        0
5        trade_allowed                     True
6         trade_expert                     True
7          margin_mode                        2
8      currency_digits                        2
9           fifo_close                    False
10             balance                  99588.3
```

© 2000-2025, MetaQuotes Ltd.

```
11              credit                          0
12              profit                     -45.13
13              equity                    99543.2
14              margin                      54.37
15         margin_free                    99488.8
16        margin_level                     183085
17      margin_so_call                         50
18        margin_so_so                         30
19      margin_initial                          0
20  margin_maintenance                          0
21              assets                          0
22         liabilities                          0
23  commission_blocked                          0
24                name                 James Smith
25              server             MetaQuotes-Demo
26            currency                        USD
27             company  MetaQuotes Software Corp.
```

더 보기

initialize, shutdown, login

© 2000-2025, MetaQuotes Ltd.

terminal_info

연결된 MetaTrader 5 클라이언트 터미널 상태 및 설정을 가져옵니다.

```
terminal_info()
```

반환 값

명명된 튜플 구조(이름 지정된 튜플)의 형식으로 정보를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

이 함수는 TerminalInfoInteger, TerminalInfoDouble 및 TerminalInfoDouble을 사용하여 한 번의 호출로 얻을 수 있는 모든 데이터를 반환합니다.

예:

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# MetaTrader 5 버전에서 데이터 표시
print(mt5.version())
# 터미널 설정 및 상태에 대한 정보 표시
terminal_info=mt5.terminal_info()
if terminal_info!=None:
    # 터미널 데이터를 '있는 그대로' 표시
    print(terminal_info)
    # 목록 형태로 데이터 표시
    print("Show terminal_info()._asdict():")
    terminal_info_dict = mt5.terminal_info()._asdict()
    for prop in terminal_info_dict:
        print("  {}={}".format(prop, terminal_info_dict[prop]))
    print()
   # 딕셔너리를 데이터프레임으로 변환하여 출력
    df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
    print("terminal_info() as dataframe:")
    print(df)

# MetaTrader 5 터미널 연결 종료
```

© 2000-2025, MetaQuotes Ltd.

```
mt5.shutdown()


결과:
MetaTrader 5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader 5 패키지 버전:  5.0.29
[500, 2366, '20 Mar 2020']
TerminalInfo(community_account=True, community_connection=True, connected=True,....
Show terminal_info()._asdict():
  community_account=True
  community_connection=True
  connected=True
  dlls_allowed=False
  trade_allowed=False
  tradeapi_disabled=False
  email_enabled=False
  ftp_enabled=False
  notifications_enabled=False
  mqid=False
  build=2366
  maxbars=5000
  codepage=1251
  ping_last=77850
  community_balance=707.10668201585
  retransmission=0.0
  company=MetaQuotes Software Corp.
  name=MetaTrader 5
  language=Russian
  path=E:\ProgramFiles\MetaTrader 5
  data_path=E:\ProgramFiles\MetaTrader 5
  commondata_path=C:\Users\Rosh\AppData\Roaming\MetaQuotes\Terminal\Common

terminal_info() as dataframe:
                속성                      값
0       community_account                   True
1    community_connection                   True
2               connected                   True
3            dlls_allowed                  False
4           trade_allowed                  False
5       tradeapi_disabled                  False
6           email_enabled                  False
7             ftp_enabled                  False
8   notifications_enabled                  False
9                    mqid                  False
10                  build                   2367
11                maxbars                   5000
12               codepage                   1251
13             ping_last                  80953
14      community_balance                707.107
```

© 2000-2025, MetaQuotes Ltd.

```
15         retransmission                   0.063593
16                company  MetaQuotes Software Corp.
17                   name                MetaTrader 5
18               language                     Russian
```

더 보기

initialize, shutdown, version

© 2000-2025, MetaQuotes Ltd.

symbols_total

MetaTrader 5 터미널에 있는 모든 금융상품의 수를 가져옵니다.

```
symbols_total()
```

반환 값

정수 값.

참고

이 함수는 SymbolsTotal()과 유사합니다. 그러나 커스텀 심볼과 MarketWatch에서 비활성화된 심볼을 포함한 모든 심볼의 수를 반환합니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 금융상품의 수 가져오기
symbols=mt5.symbols_total()
if symbols>0:
    print("Total symbols =",symbols)
else:
    print("심볼을 찾을 수 없습니다")

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

더 보기

symbols_get, symbol_select, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbols_get

MetaTrader 5 터미널에서 모든 금융상품을 가져옵니다.

```
symbols_get(
   group="GROUP"      // 심볼 선택 필터
)
```

group="GROUP"

[in] 필요한 심볼 그룹을 배열하기 위한 필터. 선택적 파라미터. 그룹이 지정된 경우 함수는 지정된 조건을 충족하는 심볼만 반환합니다.

반환 값

튜플 형태의 심볼을 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

그룹 매개 변수를 사용하면 이름을 기준으로 심볼을 정렬할 수 있습니다. '*'은 문자열의 시작과 끝에 사용될 수 있습니다

그룹 매개 변수는 명명되거나 또는 명명되지 않은채 사용할 수 있습니다. 두 옵션 모두 동일한 방식으로 작동합니다. 명명된 옵션(그룹="GROUP")을 사용하면 코드를 쉽게 읽을 수 있습니다.

그룹 매개 변수는 쉼표로 구분된 여러 조건을 포함할 수 있습니다. 조건은 '*'를 사용하여 마스크로 설정할 수 있습니다. 논리적 부정 심볼 '!'을(를) 제외에 사용할 수 있습니다. 모든 조건은 순차적으로 적용되며, 이는 그룹에 포함되는 조건 다음에 배제 조건이 먼저 지정되어야 함을 의미합니다. 예를 들면, *group*="*, !EUR"은 모든 심볼을 먼저 선택하고 그 이름에 "EUR"이 포함된 심볼을 나중에 제외해야 함을 의미합니다.

symbol_info()와는 달리, symbols_get() 함수는 단일 호출 내에서 요청된 모든 심볼에 대한 데이터를 반환합니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 모든 심볼 가져오기
symbols=mt5.symbols_get()
print('Symbols: ', len(symbols))
count=0
# 첫 다섯 개 표시
for s in symbols:
    count+=1
```

© 2000-2025, MetaQuotes Ltd.

```
    print("{}. {}".format(count,s.name))
    if count==5: break
print()

# 이름에 RU가 포함된 심볼 가져오기
ru_symbols=mt5.symbols_get("*RU*")
print('len(*RU*): ', len(ru_symbols))
for s in ru_symbols:
    print(s.name)
print()

# 이름에 USD, EUR, JPY 및 GBP가 포함되지 않은 심볼 가져오기
group_symbols=mt5.symbols_get(group="*,!*USD*,!*EUR*,!*JPY*,!*GBP*")
print('len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):', len(group_symbols))
for s in group_symbols:
    print(s.name,":",s)

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29
Symbols:  84
1. EURUSD
2. GBPUSD
3. USDCHF
4. USDJPY
5. USDCNH

len(*RU*):  8
EURUSD
USDRUB
USDRUR
EURRUR
EURRUB
FORTS.RUB.M5
EURUSD_T20
EURUSD4

len(*,!*USD*,!*EUR*,!*JPY*,!*GBP*):  13
AUDCAD : SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_dea
AUDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
AUDNZD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CADCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDCAD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDSGD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
CADMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
```

© 2000-2025, MetaQuotes Ltd.

```
CHFMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
NZDMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_c
FORTS.RTS.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FORTS.RUB.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
FOREX.CHF.M5 : SymbolInfo(custom=True, chart_mode=0, select=False, visible=False, sess
```

더 보기

symbols_total, symbol_select, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbol_info

지정된 금융상품에 대한 데이터를 가져옵니다.

```
symbol_info(
   symbol      // 금융상품명
)
```

symbol

[in] 금융상품명. 이름이 지정되지 않은 필수 매개 변수.

반환 값

명명된 튜플 구조(이름 지정된 튜플)의 형식으로 정보를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

이 함수는 SymbolInfoInteger, SymbolInfoDouble 및 SymbolInfoString을 사용하여 한 번의 호출로 얻을 수 있는 모든 데이터를 반환합니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# MarketWatch에서 EURJPY 심볼 활성화 시도
selected=mt5.symbol_select("EURJPY",True)
if not selected:
    print("EURJPY 선택 실패")
    mt5.shutdown()
    quit()

# EURJPY 심볼 속성 표시
symbol_info=mt5.symbol_info("EURJPY")
if symbol_info!=None:
    # 터미널 데이터를 '있는 그대로' 표시
    print(symbol_info)
    print("EURJPY: spread =",symbol_info.spread,"  digits =",symbol_info.digits)
    # 심볼 속성 표시 목록
    print("Show symbol_info(\"EURJPY\")._asdict():")
    symbol_info_dict = mt5.symbol_info("EURJPY")._asdict()
    for prop in symbol_info_dict:
        print("  {}={}".format(prop, symbol_info_dict[prop]))
```

© 2000-2025, MetaQuotes Ltd.

```
# MetaTrader 5 터미널 연결 종료
mt5.shutdown()


결과:
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURJPY: spread = 17   digits = 3
Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585069682
  digits=3
  spread=17
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
  option_mode=0
  option_right=0
  bid=120.024
  bidhigh=120.506
  bidlow=118.798
  ask=120.041
  askhigh=120.526
  asklow=118.828
  last=0.0
```

© 2000-2025, MetaQuotes Ltd.

```
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=0.001
trade_tick_value=0.8977708350166538
trade_tick_value_profit=0.8977708350166538
trade_tick_value_loss=0.8978272580355541
trade_tick_size=0.001
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-0.2
swap_short=-1.2
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
price_greeks_gamma=0.0
price_greeks_vega=0.0
price_greeks_rho=0.0
price_greeks_omega=0.0
price_sensitivity=0.0
basis=
category=
currency_base=EUR
currency_profit=JPY
```

© 2000-2025, MetaQuotes Ltd.

```
currency_margin=EUR
bank=
description=Euro vs Japanese Yen
exchange=
formula=
isin=
name=EURJPY
page=http://www.google.com/finance?q=EURJPY
path=Forex\EURJPY
```

더 보기

account_info, terminal_info

© 2000-2025, MetaQuotes Ltd.

symbol_info_tick

지정된 금융상품의 마지막 틱을 가져옵니다.

```
symbol_info_tick(
   symbol      // 금융상품명
)
```

symbol

[in] 금융상품명. 이름이 지정되지 않은 필수 매개 변수.

반환 값

튜플 형태로 정보를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error() 를 사용하여 얻을 수 있습니다.

참고

이 함수는 SymbolInfoTick와 유사합니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# MarketWatch에서 GBPUSD 표시 활성화 시도
selected=mt5.symbol_select("GBPUSD",True)
if not selected:
    print("GBPUSD 선택 실패")
    mt5.shutdown()
    quit()

# 최근 GBPUSD 틱 표시
lasttick=mt5.symbol_info_tick("GBPUSD")
print(lasttick)
# 목록 형식으로 틱 필드 값 표시
print("Show symbol_info_tick(\"GBPUSD\")._asdict():")
symbol_info_tick_dict = mt5.symbol_info_tick("GBPUSD")._asdict()
for prop in symbol_info_tick_dict:
    print("  {}={}".format(prop, symbol_info_tick_dict[prop]))

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

```
결과:
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29
Tick(time=1585070338, bid=1.17264, ask=1.17279, last=0.0, volume=0, time_msc=158507033
Show symbol_info_tick._asdict():
  time=1585070338
  bid=1.17264
  ask=1.17279
  last=0.0
  volume=0
  time_msc=1585070338728
  flags=2
  volume_real=0.0
```

더 보기

symbol_info, symbol_info

© 2000-2025, MetaQuotes Ltd.

symbol_select

MarketWatch 창에서 심볼을 선택하거나 창에서 심볼을 제거합니다.

```
symbol_select(
   symbol,      // 금융상품명
   enable=None  // 활성화 또는 비활성화
)
```

symbol

[in] 금융상품명. 이름이 지정되지 않은 필수 매개 변수.

enable

[in] 스위치. 이름이 지정되지 않은 매개 변수(선택 사항). 'false'인 경우 MarketWatch 창에서 심볼을 제거해야 합니다. 그렇지 않으면 MarketWatch 창에서 선택해야 합니다. 이 심볼이 있는 오픈 차트가 현재 있거나 오픈되어 있는 경우 심볼을 제거할 수 없습니다.

반환 값

성공적이면 True를 그렇지 않으면 False입니다.

참고

이 함수는SymbolSelect와 유사합니다.

예:

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
print()
# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# MarketWatch에서 EURCAD 표시 활성화 시도
selected=mt5.symbol_select("EURCAD",True)
if not selected:
    print("EURCAD 선택 실패, 오류 코드 =",mt5.last_error())
else:
    symbol_info=mt5.symbol_info("EURCAD")
    print(symbol_info)
    print("EURCAD: currency_base =",symbol_info.currency_base,"  currency_profit =",sy
    print()

    # 딕셔너리 형식으로 심볼 속성을 가져옵니다.
    print("Show symbol_info()._asdict():")
    symbol_info_dict = symbol_info._asdict()
    for prop in symbol_info_dict:
```

© 2000-2025, MetaQuotes Ltd.

```
        print("  {}={}".format(prop, symbol_info_dict[prop]))
    print()

   # 딕셔너리를 데이터프레임으로 변환하여 출력
    df=pd.DataFrame(list(symbol_info_dict.items()),columns=['property','value'])
    print("symbol_info_dict() as dataframe:")
    print(df)

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29
SymbolInfo(custom=False, chart_mode=0, select=True, visible=True, session_deals=0, ses
EURCAD: currency_base = EUR   currency_profit = CAD   currency_margin = EUR

Show symbol_info()._asdict():
  custom=False
  chart_mode=0
  select=True
  visible=True
  session_deals=0
  session_buy_orders=0
  session_sell_orders=0
  volume=0
  volumehigh=0
  volumelow=0
  time=1585217595
  digits=5
  spread=39
  spread_float=True
  ticks_bookdepth=10
  trade_calc_mode=0
  trade_mode=4
  start_time=0
  expiration_time=0
  trade_stops_level=0
  trade_freeze_level=0
  trade_exemode=1
  swap_mode=1
  swap_rollover3days=3
  margin_hedged_use_leg=False
  expiration_mode=7
  filling_mode=1
  order_mode=127
  order_gtc_mode=0
  option_mode=0
```

© 2000-2025, MetaQuotes Ltd.

```
option_right=0
bid=1.55192
bidhigh=1.55842
bidlow=1.5419800000000001
ask=1.5523099999999999
askhigh=1.55915
asklow=1.5436299999999998
last=0.0
lasthigh=0.0
lastlow=0.0
volume_real=0.0
volumehigh_real=0.0
volumelow_real=0.0
option_strike=0.0
point=1e-05
trade_tick_value=0.7043642408362214
trade_tick_value_profit=0.7043642408362214
trade_tick_value_loss=0.7044535553770941
trade_tick_size=1e-05
trade_contract_size=100000.0
trade_accrued_interest=0.0
trade_face_value=0.0
trade_liquidity_rate=0.0
volume_min=0.01
volume_max=500.0
volume_step=0.01
volume_limit=0.0
swap_long=-1.1
swap_short=-0.9
margin_initial=0.0
margin_maintenance=0.0
session_volume=0.0
session_turnover=0.0
session_interest=0.0
session_buy_orders_volume=0.0
session_sell_orders_volume=0.0
session_open=0.0
session_close=0.0
session_aw=0.0
session_price_settlement=0.0
session_price_limit_min=0.0
session_price_limit_max=0.0
margin_hedged=100000.0
price_change=0.0
price_volatility=0.0
price_theoretical=0.0
price_greeks_delta=0.0
price_greeks_theta=0.0
price_greeks_gamma=0.0
```

© 2000-2025, MetaQuotes Ltd.

```
  price_greeks_vega=0.0
  price_greeks_rho=0.0
  price_greeks_omega=0.0
  price_sensitivity=0.0
  basis=
  category=
  currency_base=EUR
  currency_profit=CAD
  currency_margin=EUR
  bank=
  description=Euro vs Canadian Dollar
  exchange=
  formula=
  isin=
  name=EURCAD
  page=http://www.google.com/finance?q=EURCAD
  path=Forex\EURCAD

symbol_info_dict() as dataframe:
        속성                                        값
0          custom                                    False
1      chart_mode                                        0
2          select                                     True
3         visible                                     True
4   session_deals                                        0
..            ...                                      ...
91        formula
92           isin
93           name                                   EURCAD
94           page  http://www.google.com/finance?q=EURCAD
95           path                               Forex\EURCAD

[96 행 x 2 열]
```

더 보기

symbol_info

© 2000-2025, MetaQuotes Ltd.

market_book_add

MetTrader 5 터미널을 지정된 심볼에 대한 Market Depth 변경 이벤트에 가입시킵니다.

```
market_book_add(
   symbol      // 금융상품명
)
```

symbol

[in] 금융상품명. 이름이 지정되지 않은 필수 매개 변수.

반환 값

성공적이면 True를 그렇지 않으면 False입니다.

참고

이 함수는 MarketBookAdd와 유사합니다.

더 보기

market_book_get, market_book_release, Market Depth structure

© 2000-2025, MetaQuotes Ltd.

market_book_get

지정된 심볼에 대한 Market Depth 항목이 포함된 BookInfo에서 튜플을 반환합니다.

```
market_book_get(
   symbol      // 금융상품명
)
```

symbol

[in] 금융상품명. 이름이 지정되지 않은 필수 매개 변수.

반환 값

주문 유형, 가격 및 대량(LOT)이 포함된 BookInfo 항목에서 Market Depth 콘텐츠를 튜플 형태로 반환합니다. BookInfo는 MqlBookInfo 구조와 유사합니다.

오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

Market Depth 변경 이벤트에 대한 가입은 market_book_add() 함수를 사용하여 사전에 수행해야 합니다.

이 함수는 MarketBookGet와 유사합니다.

예:

```
import MetaTrader5 as mt5
import time
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
print("")

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
   # MetaTrader 5 터미널 연결 종료
    mt5.shutdown()
    quit()

# EURUSD(Depth of Market)에 대한 마켓 심층 업데이트 구독
if mt5.market_book_add('EURUSD'):
  # 시장 심층 데이터를 10회 반복 수집
   for i in range(10):
        # 마켓 심층(Depth of Market) 콘텐츠 확보
        items = mt5.market_book_get('EURUSD')
        # 마켓의 전체 뎁스를 '있는 그대로' 단일 문자열로 표시
        print(items)
        # 이제 각 주문을 구분하여 표시하여 보다 명확히 확인할 수 있습니다
        if items:
```

© 2000-2025, MetaQuotes Ltd.

```
            for it in items:
                # 주문 내용
                print(it._asdict())
        # 마켓 수준 데이터의 다음 요청이 있기 전에 5초간 일시 중지
        time.sleep(5)
  # 시장 수준 업데이트(Depth of Market) 구독을 취소합니다.
   mt5.market_book_release('EURUSD')
else:
    print("mt5.market_book_add('EURUSD') failed, error code =",mt5.last_error())

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.34

(BookInfo(type=1, price=1.20038, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20038, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20032, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20023, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20017, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.2004299999999999, volume=250, volume_dbl=250.0), BookInfo(ty
{'type': 1, 'price': 1.2004299999999999, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20037, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20036, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20034, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20031, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20029, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20028, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20022, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20036, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20036, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20028, 'volume': 50, 'volume_dbl': 50.0}
```

© 2000-2025, MetaQuotes Ltd.

```
{'type': 1, 'price': 1.20026, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20035, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20035, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20029, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.20027, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20022, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20021, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20014, 'volume': 250, 'volume_dbl': 250.0}
(BookInfo(type=1, price=1.20037, volume=250, volume_dbl=250.0), BookInfo(type=1, price
{'type': 1, 'price': 1.20037, 'volume': 250, 'volume_dbl': 250.0}
{'type': 1, 'price': 1.20031, 'volume': 100, 'volume_dbl': 100.0}
{'type': 1, 'price': 1.2003, 'volume': 50, 'volume_dbl': 50.0}
{'type': 1, 'price': 1.20028, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20025, 'volume': 36, 'volume_dbl': 36.0}
{'type': 2, 'price': 1.20023, 'volume': 50, 'volume_dbl': 50.0}
{'type': 2, 'price': 1.20022, 'volume': 100, 'volume_dbl': 100.0}
{'type': 2, 'price': 1.20016, 'volume': 250, 'volume_dbl': 250.0}
```

더 보기

market_book_add, market_book_release, Market Depth structure

© 2000-2025, MetaQuotes Ltd.

market_book_release

지정된 기호에 대한 MetaTrader 5 터미널의 Market Depth 변경 이벤트 가입을 취소합니다.

```
market_book_release(
   symbol      // 금융상품명
)
```

symbol

[in] 금융상품명. 이름이 지정되지 않은 필수 매개 변수.

반환 값

성공적이면 True를 그렇지 않으면 False입니다.

참고

이 함수는 MarketBookRelease와 유사합니다.

더 보기

market_book_add, market_book_get, Market Depth structure

© 2000-2025, MetaQuotes Ltd.

copy_rates_from

지정된 날짜부터 MetaTrader 5 터미널에서 막대를 가져옵니다.

```
copy_rates_from(
   symbol,       // 심볼명
   timeframe,    // 타임프레임
   date_from,    // 초기 막대 오픈일
   count         // 막대 수
   )
```

Parameters

symbol

[in] 금융상품명, 예를 들면, "EURUSD". 이름이 지정되지 않은 필수 매개 변수.

timeframe

[in] 막대가 요청된 타임프레임. TIMEFRAME 열거값으로 값 설정. 이름이 지정되지 않은 필수 매개 변수.

date_from

[in] 요청된 샘플에서 첫 번째 막대를 연 날짜. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수.

count

[in] 수령할 막대의 수. 이름이 지정되지 않은 필수 매개 변수.

반환 값

지정된 시간, 시가, 고가, 저가, 종가, tick_volume, spread 및 real_volume 열의 numpy 어레이로 막대를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

자세한 내용은 CopyRates() 함수를 참조하십시오.

날짜가 지정된 날짜보다 작거나 같은 데이터만 반환됩니다. 즉, 막대의 오픈 시간 은 항상 지정된 막대의 오픈 시간보다 작거나 같습니다.

MetaTrader 5 터미널은 차트에서 사용자가 사용할 수 있는 내역 내의 막대를 제공합니다. 사용자가 사용할 수 있는 막대 수는 "최대. 막대, 차트에서의" 매개변수로 설정됩니다.

'datetime' 객체를 생성할 때, Python은 로컬 표준 시간대를 사용하는 반면, MetaTrader 5는 UTC 표준 시간대에 (Shift 없이) 틱과 막대 오픈 시간을 저장합니다. 따라서 시간을 사용하는 기능을 실행하기 위해 UTC 시간으로 'datetime'을 생성해야 합니다. MetaTrader 5 단말기에서 수신된 데이터의 UTC 시간이 있습니다.

TIMEFRAME 는 가능한 차트 기간 값을 포함하는 열거값입니다.

ID	Description
TIMEFRAME_M1	1 분

© 2000-2025, MetaQuotes Ltd.

ID	Description
TIMEFRAME_M2	2 분
TIMEFRAME_M3	3 분
TIMEFRAME_M4	4 분
TIMEFRAME_M5	5 분
TIMEFRAME_M6	6 분
TIMEFRAME_M10	10 분
TIMEFRAME_M12	12 분
TIMEFRAME_M12	15 분
TIMEFRAME_M20	20 분
TIMEFRAME_M30	30 분
TIMEFRAME_H1	1 시간
TIMEFRAME_H2	2 시간
TIMEFRAME_H3	3 시간
TIMEFRAME_H4	4 시간
TIMEFRAME_H6	6 시간
TIMEFRAME_H8	8 시간
TIMEFRAME_H12	12 시간
TIMEFRAME_D1	1 일
TIMEFRAME_W1	1 주
TIMEFRAME_MN1	1 달

예:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# 수집된 데이터를 표 형식으로 표시하기 위해 '수정' 모듈을 가져옵니다
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시할 최대 표 너비
# 표준 시간대 작업을 위한 pytz 모듈
import pytz
```

© 2000-2025, MetaQuotes Ltd.

```
# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 표준 시간대를 UTC로 설정
timezone = pytz.timezone("Etc/UTC")
# 로컬 시간대 오프셋 구현을 방지하기 위해 UTC 표준 시간대에 'datetime' 개체를 생성합니다.
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# UTC 표준 시간대로 2020년 1월 10일부터 10개의 EURD H4 막대를 설치
rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_H4, utc_from, 10)

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
# 수집된 데이터의 각 요소를 새 줄로 표시
print("수집된 데이터를 '있는 그대로' 표시")
for rate in rates:
    print(rate)

# 가져온 데이터로 DataFrame 생성
rates_frame = pd.DataFrame(rates)
# 시간(초)을 날짜 시간 형식으로 변환
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# 데이터 표시
print("₩n데이터와 함께 dataframe 표시")
print(rates_frame)
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

수집된 데이터를 '있는 그대로' 표시
(1578484800, 1.11382, 1.11385, 1.1111, 1.11199, 9354, 1, 0)
(1578499200, 1.11199, 1.11308, 1.11086, 1.11179, 10641, 1, 0)
(1578513600, 1.11178, 1.11178, 1.11016, 1.11053, 4806, 1, 0)
(1578528000, 1.11053, 1.11193, 1.11033, 1.11173, 3480, 1, 0)
(1578542400, 1.11173, 1.11189, 1.11126, 1.11182, 2236, 1, 0)
(1578556800, 1.11181, 1.11203, 1.10983, 1.10993, 7984, 1, 0)
(1578571200, 1.10994, 1.11173, 1.10965, 1.11148, 7406, 1, 0)
(1578585600, 1.11149, 1.11149, 1.10923, 1.11046, 7468, 1, 0)
(1578600000, 1.11046, 1.11097, 1.11033, 1.11051, 3450, 1, 0)
(1578614400, 1.11051, 1.11093, 1.11017, 1.11041, 2448, 1, 0)

데이터와 함께 dataframe 표시
                 time     open     high      low    close  tick_volume  spread  real_v
0 2020-01-08 12:00:00  1.11382  1.11385  1.11110  1.11199         9354       1
```

© 2000-2025, MetaQuotes Ltd.

1	2020-01-08 16:00:00	1.11199	1.11308	1.11086	1.11179	10641	1
2	2020-01-08 20:00:00	1.11178	1.11178	1.11016	1.11053	4806	1
3	2020-01-09 00:00:00	1.11053	1.11193	1.11033	1.11173	3480	1
4	2020-01-09 04:00:00	1.11173	1.11189	1.11126	1.11182	2236	1
5	2020-01-09 08:00:00	1.11181	1.11203	1.10983	1.10993	7984	1
6	2020-01-09 12:00:00	1.10994	1.11173	1.10965	1.11148	7406	1
7	2020-01-09 16:00:00	1.11149	1.11149	1.10923	1.11046	7468	1
8	2020-01-09 20:00:00	1.11046	1.11097	1.11033	1.11051	3450	1
9	2020-01-10 00:00:00	1.11051	1.11093	1.11017	1.11041	2448	1

더 보기

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_rates_from_pos

지정된 인덱스에서 시작하는 MetaTrader 5 터미널에서 막대 가져옵니다.

```
copy_rates_from_pos(
   symbol,       // 심볼명
   timeframe,    // 타임프레임
   start_pos,    // 초기 막대 인덱스
   count         // 막대 수
   )
```

Parameters

symbol

[in] 금융상품명, 예를 들면, "EURUSD". 이름이 지정되지 않은 필수 매개 변수.

timeframe

[in] 막대가 요청된 타임프레임. TIMEFRAME 열거값으로 값 설정. 이름이 지정되지 않은 필수 매개 변수.

start_pos

[in] 데이터를 요청하는 막대의 초기 인덱스. 막대의 번호는 현재에서 과거로 매겨집니다. 그러므로, 0 막대는 현재의 것을 의미합니다. 이름이 지정되지 않은 필수 매개 변수.

count

[in] 수령할 막대의 수. 이름이 지정되지 않은 필수 매개 변수.

반환 값

지정된 시간, 시가, 고가, 저가, 종가, tick_volume, spread 및 real_volume 열의 numpy 어레이로 막대를 반환합니다. 오류의 경우 None이 반환됩니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

자세한 내용은 CopyRates() 함수를 참조하십시오.

MetaTrader 5 터미널은 차트에서 사용자가 사용할 수 있는 내역 내의 막대를 제공합니다. 사용자가 사용할 수 있는 막대 수는 "최대. 막대, 차트에서의" 매개변수로 설정됩니다.

예:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# 수집된 데이터를 표 형식으로 표시하기 위해 '수정' 모듈을 가져옵니다
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시할 최대 표 너비
```

© 2000-2025, MetaQuotes Ltd.

```
# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# get 10 GBPUSD D1 bars from the current day
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
# 수집된 데이터의 각 요소를 새 줄로 표시
print("수집된 데이터를 '있는 그대로' 표시")
for rate in rates:
    print(rate)

# 가져온 데이터로 DataFrame 생성
rates_frame = pd.DataFrame(rates)
# 시간(초)을 날짜 시간 형식으로 변환
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# 데이터 표시
print("₩n데이터와 함께 dataframe 표시")
print(rates_frame)
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

수집된 데이터를 '있는 그대로' 표시
(1581552000, 1.29568, 1.30692, 1.29441, 1.30412, 68228, 0, 0)
(1581638400, 1.30385, 1.30631, 1.3001, 1.30471, 56498, 0, 0)
(1581897600, 1.30324, 1.30536, 1.29975, 1.30039, 49400, 0, 0)
(1581984000, 1.30039, 1.30486, 1.29705, 1.29952, 62288, 0, 0)
(1582070400, 1.29952, 1.3023, 1.29075, 1.29187, 57909, 0, 0)
(1582156800, 1.29186, 1.29281, 1.28489, 1.28792, 61033, 0, 0)
(1582243200, 1.28802, 1.29805, 1.28746, 1.29566, 66386, 0, 0)
(1582502400, 1.29426, 1.29547, 1.28865, 1.29283, 66933, 0, 0)
(1582588800, 1.2929, 1.30178, 1.29142, 1.30037, 80121, 0, 0)
(1582675200, 1.30036, 1.30078, 1.29136, 1.29374, 49286, 0, 0)

데이터와 함께 dataframe 표시
        time     open     high      low    close  tick_volume  spread  real_volume
0 2020-02-13  1.29568  1.30692  1.29441  1.30412        68228       0            0
1 2020-02-14  1.30385  1.30631  1.30010  1.30471        56498       0            0
2 2020-02-17  1.30324  1.30536  1.29975  1.30039        49400       0            0
3 2020-02-18  1.30039  1.30486  1.29705  1.29952        62288       0            0
4 2020-02-19  1.29952  1.30230  1.29075  1.29187        57909       0            0
5 2020-02-20  1.29186  1.29281  1.28489  1.28792        61033       0            0
6 2020-02-21  1.28802  1.29805  1.28746  1.29566        66386       0            0
```

© 2000-2025, MetaQuotes Ltd.

7	2020-02-24	1.29426	1.29547	1.28865	1.29283	66933	0	0
8	2020-02-25	1.29290	1.30178	1.29142	1.30037	80121	0	0
9	2020-02-26	1.30036	1.30078	1.29136	1.29374	49286	0	0

더 보기

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_rates_range

MetaTrader 5 터미널에서 지정된 날짜 범위의 막대를 가져옵니다.

```
copy_rates_range(
   symbol,       // 심볼명
   timeframe,    // 타임프레임
   date_from,    // 막대 요청 시작일
   date_to       // 막대 요청 만료일
   )
```

Parameters

symbol

[in] 금융상품명, 예를 들면, "EURUSD". 이름이 지정되지 않은 필수 매개 변수.

timeframe

[in] 막대가 요청된 타임프레임. TIMEFRAME 열거값으로 값 설정. 이름이 지정되지 않은 필수 매개 변수.

date_from

[in] 막대가 요청된 날짜. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. open time >= date_from인 막대가 반환됩니다. 이름이 지정되지 않은 필수 매개 변수.

date_to

[in] 막대 요청 만료일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. open time <= date_to인 막대가 반환됩니다. 이름이 지정되지 않은 필수 매개 변수.

반환 값

지정된 시간, 시가, 고가, 저가, 종가, tick_volume, spread 및 real_volume 열의 numpy 어레이로 막대를 반환합니다. 오류의 경우 None이 반환됩니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

자세한 내용은 CopyRates() 함수를 참조하십시오.

MetaTrader 5 터미널은 차트에서 사용자가 사용할 수 있는 내역 내의 막대를 제공합니다. 사용자가 사용할 수 있는 막대 수는 "최대. 막대, 차트에서의" 매개변수로 설정됩니다.

'datetime' 객체를 생성할 때, Python은 로컬 표준 시간대를 사용하는 반면, MetaTrader 5는 UTC 표준 시간대에 (Shift 없이) 틱과 막대 오픈 시간을 저장합니다. 따라서 시간을 사용하는 기능을 실행하기 위해 UTC 시간으로 'datetime'을 생성해야 합니다. MetaTrader 5 단말기에서 수신된 데이터의 UTC 시간이 있습니다.

예:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
```

© 2000-2025, MetaQuotes Ltd.

```
# 표에서 얻은 데이터를 표시하기 위해 'pandas' 모듈을 불러옵니다
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시할 최대 표 너비
# 표준 시간대 작업을 위한 pytz 모듈
import pytz

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 에러 코드 =",mt5.last_error())
    quit()

# set time zone to UTC
timezone = pytz.timezone("Etc/UTC")
# 로컬 표준 시간대를 구현하지 않도록 UTC 표준 시간대에 'datetime' 개체를 생성합니다 offset
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# UTC 표준 시간대에서 2020.01.10 00:00 - 2020.01.11 13:00의 간격 내에 USDJPY M5에서 막대를
rates = mt5.copy_rates_range("USDJPY", mt5.TIMEFRAME_M5, utc_from, utc_to)

# MetaTrader 5 터미널에 대한 연결 종료
mt5.shutdown()

# 수집된 데이터의 각 요소를 새 라인로 표시
print("수집된 데이터를 '있는 그대로' 표시")
counter=0
for rate in rates:
    counter+=1
    if counter<=10:
        print(rate)

# 가져온 데이터로 DataFrame 생성
rates_frame = pd.DataFrame(rates)
# 시간(초)을 'datetime' 형식으로 변환합니다.
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# display data
print("₩n데이터와 함께 dataframe 표시")
print(rates_frame.head(10))
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

수집된 데이터를 '있는 그대로' 표시
(1578614400, 109.513, 109.527, 109.505, 109.521, 43, 2, 0)
(1578614700, 109.521, 109.549, 109.518, 109.543, 215, 8, 0)
(1578615000, 109.543, 109.543, 109.466, 109.505, 98, 10, 0)
(1578615300, 109.504, 109.534, 109.502, 109.517, 155, 8, 0)
(1578615600, 109.517, 109.539, 109.513, 109.527, 71, 4, 0)
(1578615900, 109.526, 109.537, 109.484, 109.52, 106, 9, 0)
(1578616200, 109.52, 109.524, 109.508, 109.51, 205, 7, 0)
(1578616500, 109.51, 109.51, 109.491, 109.496, 44, 8, 0)
(1578616800, 109.496, 109.509, 109.487, 109.5, 85, 5, 0)
(1578617100, 109.5, 109.504, 109.487, 109.489, 82, 7, 0)
```

© 2000-2025, MetaQuotes Ltd.

```
데이터와 함께 dataframe 표시
                  time      open      high       low     close  tick_volume  spread  real_v
0  2020-01-10 00:00:00   109.513   109.527   109.505   109.521           43       2
1  2020-01-10 00:05:00   109.521   109.549   109.518   109.543          215       8
2  2020-01-10 00:10:00   109.543   109.543   109.466   109.505           98      10
3  2020-01-10 00:15:00   109.504   109.534   109.502   109.517          155       8
4  2020-01-10 00:20:00   109.517   109.539   109.513   109.527           71       4
5  2020-01-10 00:25:00   109.526   109.537   109.484   109.520          106       9
6  2020-01-10 00:30:00   109.520   109.524   109.508   109.510          205       7
7  2020-01-10 00:35:00   109.510   109.510   109.491   109.496           44       8
8  2020-01-10 00:40:00   109.496   109.509   109.487   109.500           85       5
9  2020-01-10 00:45:00   109.500   109.504   109.487   109.489           82       7
```

더 보기

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_ticks_from

지정된 날짜부터 MetaTrader 5 터미널에서 틱 가져옵니다.

```
copy_ticks_from(
   symbol,       // 심볼명
   date_from,    // 틱 요청 날짜
   count,        // 요청 틱 수
   flags         // 요청된 틱 유형을 정의하는 플래그 조합
   )
```

Parameters

symbol

[in] 금융상품명, 예를 들면, "EURUSD". 이름이 지정되지 않은 필수 매개 변수.

from

[in] 틱 요청 시작일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수.

count

[in] 수신할 틱 수. 이름이 지정되지 않은 필수 매개 변수.

flags

[in] 요청된 틱 유형을 정의하는 플래그. COPY_TICKS_INFO – 입찰 및/또는 요청 사항이 있는 틱, COPY_TICKS_TRADE – Last 및 Volume이 변경된 틱, COPY_TICKS_ALL – 모든 틱. 플래그 값은 COPY_TICKS 열거에 설명되어 있습니다. 이름이 지정되지 않은 매개 변수.

반환 값

지정된 시간, 입찰, 질문, 마지막 및 플래그 열의 숫자 배열로 틱을 반환합니다. 'flag' 값은 TICK_FLAG 열거의 플래그 조합일 수 있습니다. 오류가 발생하면 None을 반환합니다. last_error()를 사용하여 오류에 대한 정보를 얻을 수 있습니다.

참고

자세한 내용은 CopyTicks 함수를 참조하십시오.

'datetime' 객체를 생성할 때, Python은 로컬 표준 시간대를 사용하는 반면, MetaTrader 5는 UTC 표준 시간대에 (Shift 없이) 틱과 막대 오픈 시간을 저장합니다. 따라서 시간을 사용하는 기능을 실행하기 위해 UTC 시간으로 'datetime'을 생성해야 합니다. MetaTrader 5 단말기에서 수신된 데이터의 UTC 시간이 있습니다.

COPY_TICKS는 copy_ticks_from() 및 copy_ticks_range() 함수를 사용하여 요청할 수 있는 틱 유형을 정의합니다.

ID	Description
COPY_TICKS_ALL	모든 틱
COPY_TICKS_INFO	입찰 및/또는 요청 가격 변동을 포함하는 틱
COPY_TICKS_TRADE	Last 및/또는 Volume 가격 변동을 포함하는 틱

© 2000-2025, MetaQuotes Ltd.

TICK_FLAG 은 틱에 사용할 수 있는 플래그를 정의합니다. 이러한 플래그는copy_ticks_from() 및 copy_ticks_range() 함수로 얻은 틱을 설명하는 데 사용됩니다.

ID	Description
TICK_FLAG_BID	입찰가 변경
TICK_FLAG_ASK	요청가 변경
TICK_FLAG_LAST	전날 종가 변경
TICK_FLAG_VOLUME	거래량 변경
TICK_FLAG_BUY	최근 매수가 변경
TICK_FLAG_SELL	최근 매도가 변경

예:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# 수집된 데이터를 표 형식으로 표시하기 위해 '수정' 모듈을 가져옵니다
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시할 최대 표 너비
# 표준 시간대 작업을 위한 pytz 모듈
import pytz

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# set time zone to UTC
timezone = pytz.timezone("Etc/UTC")
# 로컬 시간대 오프셋 구현을 방지하기 위해 UTC 표준 시간대에 'datetime' 개체를 생성
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# UTC 표준 시간대로 2019.10.01.부터 시작하는 100 000 EURUSD 틱 요청
ticks = mt5.copy_ticks_from("EURUSD", utc_from, 100000, mt5.COPY_TICKS_ALL)
print("Ticks received:",len(ticks))

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()

# 새 라인의 각 틱에 데이터 표시
print("수집된 틱을 '있는 그대로' 표시")
count = 0
for tick in ticks:
```

© 2000-2025, MetaQuotes Ltd.

```
    count+=1
    print(tick)
    if count >= 10:
        break

# 가져온 데이터로 DataFrame 생성
ticks_frame = pd.DataFrame(ticks)
# 시간(초)을 날짜 시간 형식으로 변환
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# display data
print("₩n틱과 함께 dataframe 표시")
print(ticks_frame.head(10))
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

틱 수령: 100000
수집된 틱을 '있는 그대로' 표시
(1578614400, 1.11051, 1.11069, 0., 0, 1578614400987, 134, 0.)
(1578614402, 1.11049, 1.11067, 0., 0, 1578614402025, 134, 0.)
(1578614404, 1.1105, 1.11066, 0., 0, 1578614404057, 134, 0.)
(1578614404, 1.11049, 1.11067, 0., 0, 1578614404344, 134, 0.)
(1578614412, 1.11052, 1.11064, 0., 0, 1578614412106, 134, 0.)
(1578614418, 1.11039, 1.11051, 0., 0, 1578614418265, 134, 0.)
(1578614418, 1.1104, 1.1105, 0., 0, 1578614418905, 134, 0.)
(1578614419, 1.11039, 1.11051, 0., 0, 1578614419519, 134, 0.)
(1578614456, 1.11037, 1.11065, 0., 0, 1578614456011, 134, 0.)
(1578614456, 1.11039, 1.11051, 0., 0, 1578614456015, 134, 0.)

틱과 함께 dataframe 표시
                 time      bid      ask  last  volume        time_msc  flags  volume_re
0 2020-01-10 00:00:00  1.11051  1.11069   0.0       0  1578614400987    134          0
1 2020-01-10 00:00:02  1.11049  1.11067   0.0       0  1578614402025    134          0
2 2020-01-10 00:00:04  1.11050  1.11066   0.0       0  1578614404057    134          0
3 2020-01-10 00:00:04  1.11049  1.11067   0.0       0  1578614404344    134          0
4 2020-01-10 00:00:12  1.11052  1.11064   0.0       0  1578614412106    134          0
5 2020-01-10 00:00:18  1.11039  1.11051   0.0       0  1578614418265    134          0
6 2020-01-10 00:00:18  1.11040  1.11050   0.0       0  1578614418905    134          0
7 2020-01-10 00:00:19  1.11039  1.11051   0.0       0  1578614419519    134          0
8 2020-01-10 00:00:56  1.11037  1.11065   0.0       0  1578614456011    134          0
9 2020-01-10 00:00:56  1.11039  1.11051   0.0       0  1578614456015    134          0
```

더 보기

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

copy_ticks_range

MetaTrader 5 터미널에서 지정된 날짜 범위에 대한 틱을 가져옵니다.

```
copy_ticks_range(
   symbol,       // 심볼명
   date_from,    // 틱 요청 시작일
   date_to,      // 틱 요청 만료일
   flags         // 요청된 틱 유형을 정의하는 플래그 조합
   )
```

Parameters

symbol

[in] 금융상품명, 예를 들면, "EURUSD". 이름이 지정되지 않은 필수 매개 변수.

date_from

[in] 틱 요청 시작일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수.

date_to

[in] 틱 요청 만료일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수.

flags

[in] 요청된 틱 유형을 정의하는 플래그. COPY_TICKS_INFO – 입찰 및/또는 요청 사항이 있는 틱, COPY_TICKS_TRADE – Last 및 Volume이 변경된 틱, COPY_TICKS_ALL – 모든 틱. 플래그 값은 COPY_TICKS 열거에 설명되어 있습니다. 이름이 지정되지 않은 매개 변수.

반환 값

지정된 시간, 입찰, 질문, 마지막 및 플래그 열의 숫자 배열로 틱을 반환합니다. 'flag' 값은 TICK_FLAG 열거의 플래그 조합일 수 있습니다. 오류가 발생하면 None을 반환합니다. last_error()를 사용하여 오류에 대한 정보를 얻을 수 있습니다.

참고

자세한 내용은 CopyTicks 함수를 참조하십시오.

'datetime' 객체를 생성할 때, Python은 로컬 표준 시간대를 사용하는 반면, MetaTrader 5는 UTC 표준 시간대에 (Shift 없이) 틱과 막대 오픈 시간을 저장합니다. 따라서 시간을 사용하는 기능을 실행하기 위해 UTC 시간으로 'datetime'을 생성해야 합니다. MetaTrader 5에서 얻은 데이터는 UTC 시간이 있지만, Python은 출력을 시도할 때 로컬 시간 이동을 다시 적용합니다. 따라서 획득한 데이터도 시각적 표시를 위해 수정해야 합니다.

예:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
```

© 2000-2025, MetaQuotes Ltd.

```
# 표에서 얻은 데이터를 표시하기 위해 'pandas' 모듈을 불러옵니다
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시할 최대 표 너비
# 표준 시간대 작업을 위한 pytz 모듈
import pytz

# establish connection to MetaTrader 5 terminal
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# set time zone to UTC
timezone = pytz.timezone("Etc/UTC")
# 로컬 시간대 오프셋 구현을 방지하기 위해 UTC 표준 시간대에 'datetime' 개체를 생성합니다
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, tzinfo=timezone)
# request AUDUSD ticks within 11.01.2020 - 11.01.2020
ticks = mt5.copy_ticks_range("AUDUSD", utc_from, utc_to, mt5.COPY_TICKS_ALL)
print("Ticks received:",len(ticks))

# MetaTrader 5 터미널에 대한 연결 종료
mt5.shutdown()

# 새 라인의 각 틱에 데이터 표시
print("수집된 틱을 '있는 그대로' 표시합니다.")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# 가져온 데이터로 DataFrame 생성
ticks_frame = pd.DataFrame(ticks)
# 시간(초)을 날짜 시간 형식으로 변환
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# display data
print("₩n틱과 함께 dataframe 표시")
print(ticks_frame.head(10))
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

수령한 틱: 37008
수집된 틱을 '있는 그대로' 표시
(1578614400, 0.68577, 0.68594, 0., 0, 1578614400820, 134, 0.)
(1578614401, 0.68578, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614401, 0.68575, 0.68594, 0., 0, 1578614401128, 130, 0.)
(1578614411, 0.68576, 0.68594, 0., 0, 1578614411388, 130, 0.)
(1578614411, 0.68575, 0.68594, 0., 0, 1578614411560, 130, 0.)
(1578614414, 0.68576, 0.68595, 0., 0, 1578614414973, 134, 0.)
(1578614430, 0.68576, 0.68594, 0., 0, 1578614430188, 4, 0.)
(1578614450, 0.68576, 0.68595, 0., 0, 1578614450408, 4, 0.)
```

© 2000-2025, MetaQuotes Ltd.

```
(1578614450, 0.68576, 0.68594, 0., 0, 1578614450519, 4, 0.)
(1578614456, 0.68575, 0.68594, 0., 0, 1578614456363, 130, 0.)

틱과 함께 dataframe 표시
                 time      bid      ask  last  volume       time_msc  flags  volume_re
0 2020-01-10 00:00:00  0.68577  0.68594   0.0       0  1578614400820    134          0
1 2020-01-10 00:00:01  0.68578  0.68594   0.0       0  1578614401128    130          0
2 2020-01-10 00:00:01  0.68575  0.68594   0.0       0  1578614401128    130          0
3 2020-01-10 00:00:11  0.68576  0.68594   0.0       0  1578614411388    130          0
4 2020-01-10 00:00:11  0.68575  0.68594   0.0       0  1578614411560    130          0
5 2020-01-10 00:00:14  0.68576  0.68595   0.0       0  1578614414973    134          0
6 2020-01-10 00:00:30  0.68576  0.68594   0.0       0  1578614430188      4          0
7 2020-01-10 00:00:50  0.68576  0.68595   0.0       0  1578614450408      4          0
8 2020-01-10 00:00:50  0.68576  0.68594   0.0       0  1578614450519      4          0
9 2020-01-10 00:00:56  0.68575  0.68594   0.0       0  1578614456363    130          0
```

더 보기

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

© 2000-2025, MetaQuotes Ltd.

orders_total

활성 주문 수를 가져옵니다

```
orders_total()
```

반환 값

정수 값.

참고

이 함수는 OrdersTotal와 유사합니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 주문이 유효한지 확인
orders=mt5.orders_total()
if orders>0:
    print("Total orders=",orders)
else:
    print("주문을 찾을 수 없습니다")

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

더 보기

orders_get, positions_total

© 2000-2025, MetaQuotes Ltd.

orders_get

기호 또는 티켓별로 필터링할 수 있는 기능으로 활성 주문을 가져옵니다. 호출 옵션은 세 가지가 있습니다.

매개 변수 없이 호출. 모든 심볼에 활성 주문을 반환합니다.

```
orders_get()
```

활성 주문에 대한 심볼을 지정하는 호출이 수신되어야 합니다.

```
orders_get(
   symbol="SYMBOL"      // 심볼명
)
```

활성 주문에 대한 심볼 그룹을 지정하는 호출이 수신되어야 합니다.

```
orders_get(
   group="GROUP"        // 심볼에 대한 주문 선택 필터
)
```

주문 티켓을 지정하는 호출.

```
orders_get(
   ticket=TICKET        // 티켓
)
```

symbol="SYMBOL"

[in] 심볼명. 선택적 지명 파라미터. 심볼이 지정된 경우, *티켓* 매개 변수는 무시됩니다.

group="GROUP"

[in] 필요한 심볼 그룹을 배열하기 위한 필터. 선택적 지명 파라미터. 그룹이 지정된 경우 함수는 심볼 이름에 대해 지정된 조건을 충족하는 활성 주문만 반환합니다.

ticket=TICKET

[in] 주문 티켓 (ORDER_TICKET). 선택적으로 명명된 매개 변수.

반환 값

명명된 튜플 구조(이름 지정된 튜플)의 형식으로 정보를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

이 함수를 사용하면 OrdersTotal 및 OrderSelect tandem과 유사한 호출 한 번에 모든 활성 주문을 받을 수 있습니다.

group 매개 변수를 사용하면 심볼을 기준으로 순서를 정렬할 수 있습니다. '*'은 문자열의 시작과 끝에 사용될 수 있습니다

그룹 매개 변수는 쉼표로 구분된 여러 조건을 포함할 수 있습니다. 조건은 '*'를 사용하여 마스크로 설정할 수 있습니다. 논리적 부정 심볼 '!'을(를) 제외에 사용할 수 있습니다. 모든 조건은 순차적으로 적용되며, 이는 그룹에 포함되는 조건 다음에 배제 조건이 먼저 지정되어야 함을 의미합니다. 예를 들면, *group*="*, !EUR"은 모든 심볼에 대한 주문을 먼저 선택하고 기호 이름에 "EUR"이 포함된 주문은 나중에 제외해야 함을 의미합니다.

© 2000-2025, MetaQuotes Ltd.

예:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시될 최대 표 너비
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
print()
# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# GBPUSD에서 활성 주문의 데이터 표시
orders=mt5.orders_get(symbol="GBPUSD")
if orders is None:
    print("GBPUSD에 새 주문 없음, 에러 코드={}".format(mt5.last_error()))
else:
    print("GBPUSD에 대한 총 주문:",len(orders))
    # 모든 활성 주문 표시
    for order in orders:
        print(order)
print()

# 이름에 "*GBP*"가 포함된 심볼에 대한 주문 목록을 가져옵니다
gbp_orders=mt5.orders_get(group="*GBP*")
if gbp_orders is None:
    print("group=₩"*GBP*₩"인 주문 없음, 오류 코드={}".format(mt5.last_error()))
else:
    print("orders_get(group=\"*GBP*\")={}".format(len(gbp_orders)))
    # pandas를 이용해 이 주문들을 표로 표시. DataFrame
    df=pd.DataFrame(list(gbp_orders),columns=gbp_orders[0]._asdict().keys())
    df.drop(['time_done', 'time_done_msc', 'position_id', 'position_by_id', 'reason',
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    print(df)

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

Total orders on GBPUSD: 2
TradeOrder(ticket=554733548, time_setup=1585153667, time_setup_msc=1585153667718, time
TradeOrder(ticket=554733621, time_setup=1585153671, time_setup_msc=1585153671419, time
```

© 2000-2025, MetaQuotes Ltd.

```
orders_get(group="*GBP*")=4
      ticket           time_setup  time_setup_msc  time_expiration  type  type_time  ty
0  554733548 2020-03-25 16:27:47   1585153667718                0     3          0
1  554733621 2020-03-25 16:27:51   1585153671419                0     2          0
2  554746664 2020-03-25 16:38:14   1585154294401                0     3          0
3  554746710 2020-03-25 16:38:17   1585154297022                0     2          0
```

더 보기

orders_total, positions_get

© 2000-2025, MetaQuotes Ltd.

order_calc_margin

지정된 거래 작업을 수행하려면 계정 통화의 마진을 반환하십시오.

```
order_calc_margin(
   action,      // 주문 유형 (ORDER_TYPE_BUY 또는 ORDER_TYPE_SELL)
   symbol,      // 심볼명
   volume,      // 거래량
   price        // 시가
   )
```

Parameters

action

[in] ORDER_TYPE 열거에서 값을 가져오는 주문 유형. 이름이 지정되지 않은 필수 매개 변수.

symbol

[in] 금융상품명. 이름이 지정되지 않은 필수 매개 변수.

volume

[in] 거래 작업량. 이름이 지정되지 않은 필수 매개 변수.

price

[in] 시가. 이름이 지정되지 않은 필수 매개 변수.

반환 값

성공한 경우 실제 값, 그렇지 않은 경우 None. 오류 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

이 함수는 현재 보류 중인 주문 및 오픈 포지션을 고려하지 않고 현재 마켓 환경과 현재 계정에서 특정 주문 유형에 필요한 마진을 추정할 수 있도록 합니다. 이 함수는 OrderCalcMargin와 유사합니다.

ORDER_TYPE

ID	Description
ORDER_TYPE_BUY	시장 매수 주문
ORDER_TYPE_SELL	시장 매도 주문
ORDER_TYPE_BUY_LIMIT	구매 한도 보류 주문
ORDER_TYPE_SELL_LIMIT	판매 한도 보류 주문
ORDER_TYPE_BUY_STOP	구매 중지 보류 주문
ORDER_TYPE_SELL_STOP	판매 중지 보류 주문
ORDER_TYPE_BUY_STOP_LIMIT	주문 가격에 도달하면 주문 보류 중인 구매 한도가 StopLimit 가격에 배치됩니다
ORDER_TYPE_SELL_STOP_LIMIT	주문 가격에 도달하면 주문이 보류 중인 판매 한도가 StopLimit 가격에 배치됩니다
ORDER_TYPE_CLOSE_BY	반대위치 마감순서

© 2000-2025, MetaQuotes Ltd.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 저자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 에러 코드 =",mt5.last_error())
    quit()

# 계정 통화 가져오기
account_currency=mt5.account_info().currency
print("계정 통화:",account_currency)

# 심볼 리스트 정렬
symbols=("EURUSD","GBPUSD","USDJPY", "USDCHF","EURJPY","GBPJPY")
print("마진을 확인할 심볼:", symbols)
action=mt5.ORDER_TYPE_BUY
lot=0.1
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(심볼,"찾을 수 없음, 스킵")
        continue
    if not symbol_info.visible:
        print(심볼, "이 보이지 않습니다, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}) failed, skipped",symbol)
            continue
    ask=mt5.symbol_info_tick(symbol).ask
    margin=mt5.order_calc_margin(action,symbol,lot,ask)
    if margin != None:
        print("   {} buy {} lot margin: {} {}".format(symbol,lot,margin,account_currer
    else:
        print("order_calc_margin failed: , error code =", mt5.last_error())

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()


Result:
MetaTrader5 패키지 저자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

계정 통화: USD

마진을 체크하기 위한 심볼: ('EURUSD', 'GBPUSD', 'USDJPY', 'USDCHF', 'EURJPY', 'GBPJPY')
   EURUSD buy 0.1 lot margin: 109.91 USD
   GBPUSD buy 0.1 lot margin: 122.73 USD
   USDJPY buy 0.1 lot margin: 100.0 USD
   USDCHF buy 0.1 lot margin: 100.0 USD
   EURJPY buy 0.1 lot margin: 109.91 USD
```

© 2000-2025, MetaQuotes Ltd.

```
   GBPJPY buy 0.1 lot margin: 122.73 USD
```

더 보기

order_calc_profit, order_check

© 2000-2025, MetaQuotes Ltd.

order_calc_profit

지정된 거래 작업에 대한 계정 통화로 수익을 반환합니다.

```
order_calc_profit(
   action,          // 주문 유형 (ORDER_TYPE_BUY 또는 ORDER_TYPE_SELL)
   symbol,          // 심볼명
   volume,          // 거래량
   price_open,      // 시가
   price_close      // 종가
   );
```

Parameters

action

[in] 주문 유형에는 두개의 ORDER_TYPE 열거값 중 하나가 포함될 수 있습니다: ORDER_TYPE_BUY 또는 ORDER_TYPE_SELL. 이름이 지정되지 않은 필수 매개 변수.

symbol

[in] 금융상품명. 이름이 지정되지 않은 필수 매개 변수.

volume

[in] 거래 작업량. 이름이 지정되지 않은 필수 매개 변수.

price_open

[in] 시가. 이름이 지정되지 않은 필수 매개 변수.

price_close

[in] 종가. 이름이 지정되지 않은 필수 매개 변수.

반환 값

성공한 경우 실제 값, 그렇지 않은 경우 None. 오류 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

현재 거래 환경과 현재 계정에서 거래활동 결과를 추정하는 기능입니다. 이 함수는 OrderCalcProfit와 유사합니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 저자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 에러 코드 =",mt5.last_error())
```

© 2000-2025, MetaQuotes Ltd.

```
    quit()

# get account currency
account_currency=mt5.account_info().currency
print("계정 통화:",account_currency)

# 심볼 리스트 정렬
symbols = ("EURUSD","GBPUSD","USDJPY")
print("마진을 확인할 기호:", symbols)
# 매매이익 추정
lot=1.0
distance=300
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(심볼,"찾을 수 없음, 스킵")
        continue
    if not symbol_info.visible:
        print(심볼, "이 보이지 않습니다, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) failed, skipped",symbol)
            continue
    point=mt5.symbol_info(symbol).point
    symbol_tick=mt5.symbol_info_tick(symbol)
    ask=symbol_tick.ask
    bid=symbol_tick.bid
    buy_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_BUY,symbol,lot,ask,ask+distance*po
    if buy_profit!=None:
        print("   buy {} {} lot: profit on {} points => {} {}".format(symbol,lot,dista
    else:
        print("order_calc_profit(ORDER_TYPE_BUY) 실패, 오류 코드 =",mt5.last_error())
    sell_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_SELL,symbol,lot,bid,bid-distance*
    if sell_profit!=None:
        print("   sell {} {} lots: profit on {} points => {} {}".format(symbol,lot,dis
    else:
        print("order_calc_profit(ORDER_TYPE_SELL) failed, error code =",mt5.last_erro
    print()

# MetaTrader 5 터미널에 대한 연결 종료
mt5.shutdown()


결과:
MetaTrader5 패키지 저자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

계정 통화: USD
마진을 체크할 심볼: ('EURUSD', 'GBPUSD', 'USDJPY')
   buy EURUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell EURUSD 1.0 lot: profit on 300 points => 300.0 USD

   buy GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
   sell GBPUSD 1.0 lot: profit on 300 points => 300.0 USD
```

© 2000-2025, MetaQuotes Ltd.

```
buy USDJPY 1.0 lot: profit on 300 points => 276.54 USD
sell USDJPY 1.0 lot: profit on 300 points => 278.09 USD
```

더 보기

order_calc_margin, order_check

© 2000-2025, MetaQuotes Ltd.

order_check

필요한 거래 작업을 수행하기에 충분한 자금을 확보하십시오. 검사 결과가 MqlTradeCheckResult 구조로 반환됩니다.

```
order_check(
   request      // 요청 구조
   );
```

Parameters

request

[in] 필요한 거래 작업을 설명하는 MqlTradeRequest 유형 구조. 이름이 지정되지 않은 필수 매개 변수. 요청 및 열거 내용 작성의 예가 아래에 설명되어 있습니다.

반환 값

MqlTradeCheckResult 구조로 결과를 확인합니다. *요청* 필드는 order_check()에 전달된 거래 요청의 구조를 포함합니다.

참고

요청 전송 성공에는 요청된 거래 작업이 성공적으로 실행되지 않습니다. order_check 함수는 OrderCheck와 유사합니다.

TRADE_REQUEST_ACTIONS

ID	Description
TRADE_ACTION_DEAL	지정된 매개 변수를 사용하여 즉시 거래 주문(시장 주문 설정)
TRADE_ACTION_PENDING	지정된 조건에서 거래를 수행하기 위한 주문(보류 주문)
TRADE_ACTION_SLTP	스탑 로스 및 이익 실현의 오픈 포지션 변경
TRADE_ACTION_MODIFY	이전에 실행된 거래 주문의 매개 변수 변경
TRADE_ACTION_REMOVE	이전에 보류 중인 주문 제거
TRADE_ACTION_CLOSE_BY	반대로 포지션 닫음

ORDER_TYPE_FILLING

ID	Description
ORDER_FILLING_FOK	이 실행 정책은 지정된 볼륨에서만 주문을 실행할 수 있음을 의미합니다. 현재 마켓에서 필요한 금융상품을 구할 수 없는 경우 주문이 실행되지 않습니다. 원하는 거래량은 사용 가능한 여러 오퍼로 구성할 수 있습니다.
ORDER_FILLING_IOC	주문에 지정된 거래량 내에서 시장에서 사용 가능한 최대 거래량으로 거래를 실행하기로 합의합니다. 요청을 완전히 작성할 수 없는 경우 사용 가능한 거래량으로 주문이 실행되고 나머지 거래량이 취소됩니다.

© 2000-2025, MetaQuotes Ltd.

ID	Description
ORDER_FILLING_RETURN	이 정책은 마켓에만 사용됩니다. (ORDER_TYPE_BUY 및 ORDER_TYPE_SELL), 제한 및 제한정지 주문 (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT 및 ORDER_TYPE_SELL_STOP_LIMIT) 그리고 마켓 또는 거래소 실행 모드의 심볼에만 해당됩니다. 일부만 채워지면 남은 수량의 시장 또는 한정 주문이 취소되지 않고 추가로 처리됩니다. ORDER_TYPE_BUY_STOP_LIMIT 및 ORDER_TYPE_SELL_STOP_LIMIT 주문을 활성화하는 동안, ORDER_FILLING_RETURN 유형의 적절한 제한 주문 ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT가 생성됩니다.

ORDER_TYPE_TIME

ID	Description
ORDER_TIME_GTC	주문이 수동으로 취소될 때까지 대기열에 남아 있습니다
ORDER_TIME_DAY	주문이 현재 거래일 동안만 활성 상태입니다
ORDER_TIME_SPECIFIED	주문이 지정된 날짜까지 활성 상태입니다.
ORDER_TIME_SPECIFIED_DAY	주문은 지정된 날짜의 23:59:59까지 활성화됩니다. 이 시간이 거래 세션을 벗어난 것으로 보이면 만료가 가장 가까운 거래 시간에 처리됩니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 저자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 계정 통화 가져오기
account_currency=mt5.account_info().currency
print("계정 통화:",account_currency)

# 요청 구조 준비
symbol="USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(심볼, "찾을 수 없음, order_check() 호출 불가")
    mt5.shutdown()
    quit()

# 심볼을 MarketWatch에서 사용할 수 없는 경우 추가
if not symbol_info.visible:
```

© 2000-2025, MetaQuotes Ltd.

```
    print(심볼, "보이지 않음, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}) failed, exit",symbol)
        mt5.shutdown()
        quit()

# 요청 준비
point=mt5.symbol_info(symbol).point
request = {
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": 1.0,
    "type": mt5.ORDER_TYPE_BUY,
    "price": mt5.symbol_info_tick(symbol).ask,
    "sl": mt5.symbol_info_tick(symbol).ask-100*point,
    "tp": mt5.symbol_info_tick(symbol).ask+100*point,
    "deviation": 10,
    "magic": 234000,
    "comment": "python script",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# 점검을 수행하고 '있는 그대로' 결과를 표시합니다.
result = mt5.order_check(request)
print(result);
# 결과를 딕셔너리로 요청하여 요소별로 표시
result_dict=result._asdict()
for field in result_dict.keys():
    print("   {}={}".format(field,result_dict[field]))
    # 거래 요청 구조인 경우 요소별로도 표시
    if field=="request":
        traderequest_dict=result_dict[field]._asdict()
        for tradereq_filed in traderequest_dict:
            print("       traderequest: {}={}".format(tradereq_filed,traderequest_dict

# MetaTrader 5 터미널에 대한 연결 종료
mt5.shutdown()
```

결과:

```
MetaTrader5 패키지 저자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

계정 통화: USD
   retcode=0
   balance=101300.53
   equity=68319.53
   profit=-32981.0
   margin=51193.67
   margin_free=17125.86
   margin_level=133.45308121101692
   comment=Done
   request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=1.0,
```

© 2000-2025, MetaQuotes Ltd.

```
        traderequest: action=1
        traderequest: magic=234000
        traderequest: order=0
        traderequest: symbol=USDJPY
        traderequest: volume=1.0
        traderequest: price=108.081
        traderequest: stoplimit=0.0
        traderequest: sl=107.98100000000001
        traderequest: tp=108.181
        traderequest: deviation=10
        traderequest: type=0
        traderequest: type_filling=2
        traderequest: type_time=0
        traderequest: expiration=0
        traderequest: comment=python script
        traderequest: position=0
        traderequest: position_by=0
```

더 보기

order_send, OrderCheck, Trading operation types, Trading request structure, Structure of the trading request check results, Structure of the trading request result

© 2000-2025, MetaQuotes Ltd.

order_send

터미널에서 거래 서버로 요청 : 거래 작업을 수행하기 위한, 을 보냅니다. 이 함수는 OrderSend와 유사합니다.

```
order_send(
   request      // 요청 구조
   );
```

Parameters

request

[in] 필요한 거래 작업을 설명하는 MqlTradeRequest 유형 구조. 이름이 지정되지 않은 필수 매개 변수. 요청 및 열거 내용 작성의 예가 아래에 설명되어 있습니다.

반환 값

MqlTradeResult 구조로의 실행 결과. 응답의 *요청* 필드에는 order_send()에 전달된 거래 요청 구조가 포함되어 있습니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

MqlTradeRequest 거래 요청 구조

Field	Description
action	거래 작업 유형. 이 값은 TRADE_REQUEST_ACTIONS 열거값 중 하나일 수 있습니다
magic	EA ID. 거래 주문의 분석 처리를 수행할 수 있습니다. 거래 요청 전송 시 각 EA에서 고유 ID를 설정할 수 있습니다
order	주문 티켓. 보류 중인 주문을 수정하는 데 필요
symbol	주문이 배치된 거래 상품의 이름. 주문 수정 및 마감 포지션에는 필요하지 않음
volume	요청된 거래량(lot). 실제 거래량은 주문 실행 유형에 따라 달라집니다.
price	주문을 실행할 가격. "마켓 실행" (SYMBOL_TRADE_EXECUTION_MARKET) 유형의 계측기에 대해 TRADE_ACTION_DEAL 유형을 보유한 시장 주문의 경우 가격이 설정되지 않습니다
stoplimit	보류중 Limit 주문 당 가격이 'price' 값에 도달하면 설정됩니다(이 조건은 필수). 보류 중인 주문이 해당 시점까지 거래 시스템으로 전달되지 않습니다.
sl	A price a Stop Loss 주문은 가격이 좋지 않은 방향으로 움직일 때 활성화됩니다
tp	A price a Take Profit 주문은 가격이 유리한 방향으로 움직일 때 활성화됩니다
deviation	요청된 가격에서 허용되는 최대 편차, 다음에 지정됨:포인트
type	주문 유형. 이 값은 ORDER_TYPE 열거값 중 하나일 수 있습니다
type_filling	주문 작성 유형. 이 값은 ORDER_TYPE_FILLING 값들 중 하나일 수 있습니다
type_time	만기별 주문유형. 이 값은 ORDER_TYPE_TIME 값들 중 하나일 수 있습니다
expiration	보류 주문 만료 시간 (TIME_SPECIFIED 유형 주문용)
comment	주문에 코멘트달기

© 2000-2025, MetaQuotes Ltd.

Field	Description
position	Position ticket. 포지션을 변경하고 닫을 때 명확한 식별을 위해 포지션을 채웁니다. 보통, 그것은 그 포지션을 오픈한 주문의 티켓과 같습니다.
position_by	반대 포지션 티켓. 반대쪽(같은 기호이지만 반대 방향으로 오픈)으로 포지션을 닫을 때 사용됩니다.

참고

거래 요청은 거래 서버에서 여러 검증 단계를 통과합니다. 먼저 필요한 모든 *요청* 필드의 유효성을 확인합니다. 오류가 없으면, 서버는 추가 처리를 위해 주문을 수락합니다. 거래 작업 실행에 대한 자세한 내용은 OrderSend 함수 설명을 참조하십시오.

예:

```
import time
import MetaTrader5 as mt5

# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 저자: ", mt5.__author__)
print("MetaTrader5 패키지 버전: ", mt5.__version__)

# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 매수 요청 구조를 준비
symbol = "USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(심볼, "찾을 수 없음, order_check() 호출 불가")
    mt5.shutdown()
    quit()

# 심볼을 MarketWatch에서 사용할 수 없는 경우 추가
if not symbol_info.visible:
    print(심볼, "보이지 않음, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) failed, exit",symbol)
        mt5.shutdown()
        quit()

lot = 0.1
point = mt5.symbol_info(symbol).point
price = mt5.symbol_info_tick(symbol).ask
deviation = 20
request = {
    "action": mt5.TRADE_ACTION_DEAL,
```

© 2000-2025, MetaQuotes Ltd.

```
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_BUY,
    "price": price,
    "sl": price - 100 * point,
    "tp": price + 100 * point,
    "deviation": deviation,
    "magic": 234000,
    "comment": "python script open",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}

# 거래 요청 전송
result = mt5.order_send(request)
# 실행 결과 확인
print("1. order_send(): by {} {} lots at {} with deviation={} points".format(symbol,lo
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("2. order_send failed, retcode={}".format(result.retcode))
   # 결과를 사전으로 요청하여 요소별로 표시
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # 거래 요청 구조인 경우 요소별로도 표시
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_
    print("shutdown() and quit")
    mt5.shutdown()
    quit()

print("2. order_send done, ", result)
print("   opened position with POSITION_TICKET={}".format(result.order))
print("   sleep 2 seconds before closing position #{}".format(result.order))
time.sleep(2)
# 마감 요청 생성
position_id=result.order
price=mt5.symbol_info_tick(symbol).bid
deviation=20
request={
    "action": mt5.TRADE_ACTION_DEAL,
    "symbol": symbol,
    "volume": lot,
    "type": mt5.ORDER_TYPE_SELL,
    "position": position_id,
    "price": price,
    "deviation": deviation,
    "magic": 234000,
```

© 2000-2025, MetaQuotes Ltd.

```
    "comment": "python script close",
    "type_time": mt5.ORDER_TIME_GTC,
    "type_filling": mt5.ORDER_FILLING_RETURN,
}
# 거래 요청 전송
result=mt5.order_send(request)
# 실행 결과 확인
print("3. close position #{}: sell {} {} lots at {} with deviation={} points".format(p
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("4. order_send failed, retcode={}".format(result.retcode))
    print("   result",result)
else:
    print("4. position #{} closed, {}".format(position_id,result))
   # 결과를 사전으로 요청하여 요소별로 표시
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # 거래 요청 구조인 경우 요소별로도 표시
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()

Result
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29
1. order_send(): by USDJPY 0.1 lots at 108.023 with deviation=20 points
2. order_send done,  OrderSendResult(retcode=10009, deal=535084512, order=557416535, v
   opened position with POSITION_TICKET=557416535
   sleep 2 seconds before closing position #557416535
3. close position #557416535: sell USDJPY 0.1 lots at 108.018 with deviation=20 points
4. position #557416535 closed, OrderSendResult(retcode=10009, deal=535084631, order=55
   retcode=10009
   deal=535084631
   order=557416654
   volume=0.1
   price=108.015
   bid=108.015
   ask=108.02
   comment=Request executed
   request_id=55
   retcode_external=0
   request=TradeRequest(action=1, magic=234000, order=0, symbol='USDJPY', volume=0.1,
       traderequest: action=1
       traderequest: magic=234000
       traderequest: order=0
```

© 2000-2025, MetaQuotes Ltd.

```
        traderequest: symbol=USDJPY
        traderequest: volume=0.1
        traderequest: price=108.018
        traderequest: stoplimit=0.0
        traderequest: sl=0.0
        traderequest: tp=0.0
        traderequest: deviation=20
        traderequest: type=1
        traderequest: type_filling=2
        traderequest: type_time=0
        traderequest: expiration=0
        traderequest: comment=python script close
        traderequest: position=557416535
        traderequest: position_by=0
```

더 보기

order_check, OrderSend,Trading operation types, Trading request structure, Structure of the trading request check results, Structure of the trading request result

© 2000-2025, MetaQuotes Ltd.

positions_total

오픈 포지션 수를 가져옵니다.

```
positions_total()
```

반환 값

정수 값.

참고

이 함수는 PositionsTotal와 유사합니다.

예:

```
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 오픈 포지션 존재 유무 확인
positions_total=mt5.positions_total()
if positions_total>0:
    print("Total positions=",positions_total)
else:
    print("포지션을 찾을 수 없습니다")

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

더 보기

positions_get, orders_total

© 2000-2025, MetaQuotes Ltd.

positions_get

기호 또는 티켓으로 필터링할 수 있는 오픈 포지션을 가져옵니다. 호출 옵션은 세 가지가 있습니다.

매개 변수 없이 호출. 모든 심볼에 대한 오픈 포지션을 반환합니다.

```
positions_get()
```

오픈 포지션을 지정하는 호출은 다음에 대해 수신되어야 합니다.

```
positions_get(
   symbol="SYMBOL"      // 심볼명
)
```

오픈 포지션의 심볼 그룹을 지정하는 호출은 다음에 대해 수신되어야 합니다.

```
positions_get(
   group="GROUP"        // 심볼별 위치 선택을 위한 필터
)
```

포지션 티켓을 지정하는 호출.

```
positions_get(
   ticket=TICKET        // 티켓
)
```

Parameters

symbol="SYMBOL"

[in] 심볼명. 선택적 지명 파라미터. 심볼이 지정된 경우, *티켓* 매개 변수는 무시됩니다.

group="GROUP"

[in] 필요한 심볼 그룹을 배열하기 위한 필터. 선택적 지명 파라미터. 그룹이 지정된 경우 함수는 심볼명에 대해 지정된 기준을 충족하는 포지션만 반환합니다.

ticket=TICKET

[in] 포지션 티켓 (POSITION_TICKET). 선택적으로 명명된 매개 변수.

반환 값

명명된 튜플 구조(이름 지정된 튜플)의 형식으로 정보를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

이 함수를 사용하면 PositionsTotal 및 PositionSelect tandem과 유사한 한 호출 내에서 모든 오픈 포지션을 수신할 수 있습니다.

그룹 매개 변수는 쉼표로 구분된 여러 조건을 포함할 수 있습니다. 조건은 '*'를 사용하여 마스크로 설정할 수 있습니다. 논리적 부정 심볼 '!'을(를) 제외에 사용할 수 있습니다. 모든 조건은 순차적으로 적용되며, 이는 그룹에 포함되는 조건 다음에 배제 조건이 먼저 지정되어야 함을 의미합니다. 예를 들면, *group*="*, !EUR"은 모든 심볼에 대한 포지션을 먼저 선택하고 심볼 이름에 "EUR"이 포함된 거래는 나중에 제외해야 함을 의미합니다.

예:

© 2000-2025, MetaQuotes Ltd.

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시될 최대 표 너비
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
print()
# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# USDCHF에서의 오픈 포지션 가져오기
positions=mt5.positions_get(symbol="USDCHF")
if positions==None:
    print("USDCHF에 포지션 없음, 오류 코드={}".format(mt5.last_error()))
elif len(positions)>0:
    print("USDCHF에서의 총 포지션 =",len(positions))
    # 모든 오픈 포지션 표시
    for position in positions:
        print(position)

# 이름에 "*USD*"가 포함된 심볼의 포지션 목록을 가져옵니다
usd_positions=mt5.positions_get(group="*USD*")
if usd_positions==None:
    print("No positions with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(usd_positions)>0:
    print("positions_get(group=\"*USD*\")={}".format(len(usd_positions)))
    # 이러한 포지션을 pandas.DataFrame을 사용해 표로 표시
    df=pd.DataFrame(list(usd_positions),columns=usd_positions[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    df.drop(['time_update', 'time_msc', 'time_update_msc', 'external_id'], axis=1, inp
    print(df)

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

positions_get(group="*USD*")=5
      ticket                time  type  magic  identifier  reason  volume  price_open
0  548297723 2020-03-18 15:00:55     1      0   548297723       3    0.01     1.09301
1  548655158 2020-03-18 20:31:26     0      0   548655158       3    0.01     1.08676
2  548663803 2020-03-18 20:40:04     0      0   548663803       3    0.01     1.08640
3  548847168 2020-03-19 01:10:05     0      0   548847168       3    0.01     1.09545
```

© 2000-2025, MetaQuotes Ltd.

4	548847194	2020-03-19 01:10:07	0	0	548847194	3	0.02	1.09536

더 보기

positions_total, orders_get

© 2000-2025, MetaQuotes Ltd.

history_orders_total

지정된 간격 내에 거래 내역의 주문 수를 가져옵니다.

```
history_orders_total(
   date_from,     // 주문 요청 시작일
   date_to        // 주문 요청 만료일
   )
```

Parameters

date_from

[in] 주문 요청 시작일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수.

date_to

[in] 주문 요청 만료일 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수.

반환 값

정수 값.

참고

이 함수는 HistoryOrdersTotal와 유사합니다.

예:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 내역에서 주문 수를 가져옵니다
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_total(from_date, datetime.now())
if history_orders>0:
    print("Total history orders=",history_orders)
else:
    print("주문을 내역에서 찾을 수 없습니다")

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

더 보기

history_orders_get, history_deals_total

© 2000-2025, MetaQuotes Ltd.

history_orders_get

티켓 또는 포지션으로 필터링할 수 있는 함수로, 거래 내역에서 주문을 가져옵니다. 호출 옵션은 세 가지가 있습니다.

시간 간격을 지정하여 호출합니다. 지정된 간격 내에 해당하는 모든 주문을 반환합니다.

```
history_orders_get(
   date_from,                 // 주문 요청 시작일
   date_to,                   // 주문 요청 만료일
   group="GROUP"          // 심볼별 주문 선택을 위한 필터
   )
```

주문 티켓을 지정하는 호출. 지정된 티켓으로 주문을 반환합니다.

```
history_orders_get(
   ticket=TICKET          // 주문 티켓
)
```

포지션 티켓을 지정해 호출합니다. ORDER_POSITION_ID 속성에 지정된 포지션 티켓으로 모든 주문을 반환합니다.

```
history_orders_get(
   position=POSITION     // 포지션 티켓
)
```

Parameters

date_from

[in] 주문 요청 시작일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수가 먼저 지정됩니다.

date_to

[in] 주문 요청 만료일 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수가 초 단위로 지정됩니다.

group="GROUP"

[in] 필요한 심볼 그룹을 배열하기 위한 필터. 선택적 지명 파라미터. 그룹이 지정된 경우 함수는 심볼 이름에 대해 지정된 조건을 충족하는 주문만 반환합니다.

ticket=TICKET

[in] 받아야 할 주문 티켓. 선택적 파라미터. 지정하지 않으면 필터가 적용되지 않습니다.

position=POSITION

[in] 주문은 모든 포지션(ORDER_POSITION_ID에 저장됨)의 티켓을 수령해야 함. 선택적 파라미터. 지정하지 않으면 필터가 적용되지 않습니다.

반환 값

명명된 튜플 구조(이름 지정된 튜플)의 형식으로 정보를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

참고

© 2000-2025, MetaQuotes Ltd.

이 함수를 사용하면 HistoryOrdersTotal 및 HistoryOrderSelect 탠덤과 유사한 단일 요청에서 지정된 기간 내에 모든 주문 내역을 받을 수 있습니다.

그룹 매개 변수는 쉼표로 구분된 여러 조건을 포함할 수 있습니다. 조건은 '*'를 사용하여 마스크로 설정할 수 있습니다. 논리적 부정 심볼 '!'을(를) 제외에 사용할 수 있습니다. 모든 조건은 순차적으로 적용되며, 이는 그룹에 포함되는 조건 다음에 배제 조건이 먼저 지정되어야 함을 의미합니다. 예를 들면, *group*="*, !EUR"은 모든 심볼에 대한 거래를 먼저 선택하고 심볼 이름에 "EUR"이 포함된 거래는 나중에 제외해야 함을 의미합니다.

예:

```
from datetime import datetime
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시될 최대 표 너비
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
print()
# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 내역에서 주문 수를 가져옵니다
from_date=datetime(2020,1,1)
to_date=datetime.now()
history_orders=mt5.history_orders_get(from_date, to_date, group="*GBP*")
if history_orders==None:
    print("No history orders with group=\"*GBP*\", error code={}".format(mt5.last_erro
elif len(history_orders)>0:
    print("history_orders_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,le
print()

# 모든 내역상 주문을 포지션별로 표시
position_id=530218319
position_history_orders=mt5.history_orders_get(position=position_id)
if position_history_orders==None:
    print("No orders with position #{}".format(position_id))
    print("error code =",mt5.last_error())
elif len(position_history_orders)>0:
    print("Total history orders on position #{}: {}".format(position_id,len(position_h
   # 지정된 포지션을 가지고 있는 모든 과거 주문을 표시
    for position_order in position_history_orders:
        print(position_order)
    print()
    # pandas를 이용해 이 주문들을 표로 표시. DataFrame
    df=pd.DataFrame(list(position_history_orders),columns=position_history_orders[0]._
    df.drop(['time_expiration','type_time','state','position_by_id','reason','volume_
```

© 2000-2025, MetaQuotes Ltd.

```
    df['time_setup'] = pd.to_datetime(df['time_setup'], unit='s')
    df['time_done'] = pd.to_datetime(df['time_done'], unit='s')
    print(df)

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()

결과:
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

history_orders_get(2020-01-01 00:00:00, 2020-03-25 17:17:32.058795, group="*GBP*")=14

포지션 #530218319의 총 주문 내역: 2
TradeOrder(ticket=530218319, time_setup=1582282114, time_setup_msc=1582282114681, time
TradeOrder(ticket=535548147, time_setup=1583176242, time_setup_msc=1583176242265, time

      ticket           time_setup  time_setup_msc           time_done  time_done_msc  t
0  530218319 2020-02-21 10:48:34   1582282114681 2020-02-21 16:49:37  1582303777582
1  535548147 2020-03-02 19:10:42   1583176242265 2020-03-02 19:10:42  1583176242265
```

더 보기

history_deals_total, history_deals_get

© 2000-2025, MetaQuotes Ltd.

history_deals_total

지정된 간격 내에 거래 내역의 딜 수를 가져옵니다.

```
history_deals_total(
   date_from,    // 딜 요청 시작일
   date_to       // 딜 요청 만료일
   )
```

Parameters

date_from

[in] 딜 요청 시작일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수.

date_to

[in] 딜 요청 만료일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수.

반환 값

정수 값.

참고

이 함수는 HistoryDealsTotal와 유사합니다.

예:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)

# MetaTrader 5 터미널 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 내역에서 딜 수 가져오기
from_date=datetime(2020,1,1)
to_date=datetime.now()
deals=mt5.history_deals_total(from_date, to_date)
if deals>0:
    print("총 딜=",deals)
else:
    print("내역에서 딜을 찾을 수 없음")

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

© 2000-2025, MetaQuotes Ltd.

더 보기

history_deals_get, history_orders_total

© 2000-2025, MetaQuotes Ltd.

history_deals_get

티켓 또는 위치별로 필터링할 수 있는 기능을 사용하여 지정된 간격 내에 거래 내역에서 거래를 가져옵니다.

시간 간격을 지정하여 호출합니다. 지정된 간격 내에 해당하는 모든 거래를 반환합니다.

```
history_deals_get(
   date_from,                 // 딜 요청 시작일
   date_to,                   // 딜 요청 만료일
   group="GROUP"          // 심볼에 대한 거래 선택 필터
   )
```

주문 티켓을 지정하기 위해 요청. DEAL_ORDER 속성에 지정된 주문 티켓이 있는 모든 거래를 반환합니다.

```
history_deals_get(
   ticket=TICKET          // 주문 티켓
)
```

포지션 티켓을 지정해 호출합니다. DEAL_POSITION_ID 속성에 지정된 포지션 티켓이 있는 모든 거래를 반환합니다.

```
history_deals_get(
   position=POSITION      // 포지션 티켓
)
```

Parameters

date_from

[in] 주문 요청 시작일. 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수가 먼저 지정됩니다.

date_to

[in] 주문 요청 만료일 'datetime' 개체에 의해 또는 1970.01.01 이후 경과된 시간(초)으로 설정됨. 이름이 지정되지 않은 필수 매개 변수가 초 단위로 지정됩니다.

group="GROUP"

[in] 필요한 심볼 그룹을 배열하기 위한 필터. 선택적 지명 파라미터. 그룹이 지정된 경우 함수는 심볼명에 대해 지정된 조건을 충족하는 거래만 반환합니다.

ticket=TICKET

[in] 다음에 대한 모든 거래가 접수되어야 하는 주문(DEAL_ORDER에 저장됨)의 티켓. 선택적 파라미터. 지정하지 않으면 필터가 적용되지 않습니다.

position=POSITION

[in] 모든 거래를 수집해야 하는 포지션(DEAL_POSITION_ID에 저장됨) 티켓. 선택적 파라미터. 지정하지 않으면 필터가 적용되지 않습니다.

반환 값

명명된 튜플 구조(이름 지정된 튜플)의 형식으로 정보를 반환합니다. 오류가 발생하면 None을 반환합니다. 오류에 대한 정보는 last_error()를 사용하여 얻을 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

참고

이 함수를 사용하면 HistoryDealsTotal 및 HistoryDealSelect 탠덤과 유사한 단일 요청에서 지정된 기간 내의 모든 거래 이력을 받을 수 있습니다.

그룹 매개 변수를 사용해 심볼을 기준으로 거래를 정렬할 수 있습니다. '*'은 문자열의 시작과 끝에 사용될 수 있습니다

그룹 매개 변수는 쉼표로 구분된 여러 조건을 포함할 수 있습니다. 조건은 '*'를 사용하여 마스크로 설정할 수 있습니다. 논리적 부정 심볼 '!'을(를) 제외에 사용할 수 있습니다. 모든 조건은 순차적으로 적용되며, 이는 그룹에 포함되는 조건 다음에 배제 조건이 먼저 지정되어야 함을 의미합니다. 예를 들면, *group*="*, !EUR"은 모든 심볼에 대한 거래를 먼저 선택하고 심볼 이름에 "EUR"이 포함된 거래는 나중에 제외해야 함을 의미합니다.

예:

```
import MetaTrader5 as mt5
from datetime import datetime
import pandas as pd
pd.set_option('display.max_columns', 500) # 표시될 칼럼 수
pd.set_option('display.width', 1500)      # 표시될 최대 표 너비
# MetaTrader 5 패키지에 데이터 표시
print("MetaTrader5 패키지 작성자: ",mt5.__author__)
print("MetaTrader5 패키지 버전: ",mt5.__version__)
print()
# MetaTrader 5 터미널과의 연결 설정
if not mt5.initialize():
    print("initialize() 실패, 오류 코드 =",mt5.last_error())
    quit()

# 내역에서 딜 수 가져오기
from_date=datetime(2020,1,1)
to_date=datetime.now()
# 이름에 지정된 간격 내에 "GBP"가 포함된 심볼에 대한 딜 가져오기
deals=mt5.history_deals_get(from_date, to_date, group="*GBP*")
if deals==None:
    print("No deals with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(deals)> 0:
    print("history_deals_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,len

# 이름에 "EUR" 또는 "GBP"가 포함되지 않은 심볼에 대한 거래를 가져오기
deals = mt5.history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*")
if deals == None:
    print("딜이 없습니다, 에러 코드={}".format(mt5.last_error()))
elif len(deals) > 0:
    print("history_deals_get(from_date, to_date, group=\"*,!*EUR*,!*GBP*\") =", len(de
    # 수집된 딜을 '있는 그대로' 표시
    for deal in deals:
        print("  ",deal)
    print()
    # pandas.DataFrame을 사용하여 테이블로써 이러한 딜을 표시
```

© 2000-2025, MetaQuotes Ltd.

```
    df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)
print("")

# 포지션 #530218319와 관련한 딜 가져오기
position_id=530218319
position_deals = mt5.history_deals_get(position=position_id)
if position_deals == None:
    print("No deals with position #{}".format(position_id))
    print("error code =", mt5.last_error())
elif len(position_deals) > 0:
    print("Deals with position id #{}: {}".format(position_id, len(position_deals)))
    # pandas.DataFrame을 사용하여 테이블로써 이러한 딜을 표시
    df=pd.DataFrame(list(position_deals),columns=position_deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)

# MetaTrader 5 터미널 연결 종료
mt5.shutdown()
```

결과:

```
MetaTrader5 패키지 작성자:  MetaQuotes Software Corp.
MetaTrader5 패키지 버전:  5.0.29

history_deals_get(from_date, to_date, group="*GBP*") = 14

history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*") = 7
   TradeDeal(ticket=506966741, order=0, time=1582202125, time_msc=1582202125419, type=
   TradeDeal(ticket=507962919, order=530218319, time=1582303777, time_msc=158230377758
   TradeDeal(ticket=513149059, order=535548147, time=1583176242, time_msc=158317624226
   TradeDeal(ticket=516943494, order=539349382, time=1583510003, time_msc=158351000389
   TradeDeal(ticket=516943915, order=539349802, time=1583510025, time_msc=158351002505
   TradeDeal(ticket=517139682, order=539557870, time=1583520201, time_msc=158352020122
   TradeDeal(ticket=517139716, order=539557909, time=1583520202, time_msc=158352020297

      ticket      order                time       time_msc  type  entry  magic  positi
0  506966741          0 2020-02-20 12:35:25  1582202125419     2      0      0
1  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    5302
2  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
3  516943494  539349382 2020-03-06 15:53:23  1583510003895     1      0      0    5393
4  516943915  539349802 2020-03-06 15:53:45  1583510025054     0      0      0    5393
5  517139682  539557870 2020-03-06 18:43:21  1583520201227     0      1      0    5393
6  517139716  539557909 2020-03-06 18:43:22  1583520202971     1      1      0    5393

Deals with position id #530218319: 2
      ticket      order                time       time_msc  type  entry  magic  positi
0  507962919  530218319 2020-02-21 16:49:37  1582303777582     0      0      0    5302
1  513149059  535548147 2020-03-02 19:10:42  1583176242265     1      1      0    5302
```

© 2000-2025, MetaQuotes Ltd.

더 보기

history_deals_total, history_orders_get

© 2000–2025, MetaQuotes Ltd.

머신러닝에서 ONNX 모델

ONNX (Open Neural Network Exchange)는 머신러닝 모델을 위한 오픈 소스 형식입니다. 이 프로젝트에는 다음과 같은 몇 가지 주요 이점이 있습니다.

- ONNX는 Microsoft, Facebook, Amazon 및 기타 파트너와 같은 거대 기업의 지원을 받습니다.
- ONNX의 개방성은 서로 다른 기계 학습 툴킷 간에 형식 변환을 가능하게 합니다. 가령 Microsoft의 ONNXML도구가 모델을 ONNX 형식으로 변환하도록 할 수 있습니다.
- MQL5는 전달된 매개변수 유형이 모델과 일치하지 않는 경우 모델의 입력 및 출력을 위한 자동 데이터 유형 변환을 제공합니다. .
- ONNX 모델은 다양한 머신러닝 도구를 사용하여 만들 수 있습니다. 현재 Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch 및 OpenCV에서 지원됩니다. 다른 인기 있는 프레임워크 및 라이브러리에 대한 인터페이스도 사용할 수 있습니다.
- MQL5 언어를 사용하면 거래 전략에서 ONNX 모델을 구현할 수 있습니다. 금융 시장에서 효율적인 운영을 위해 MetaTrader 5 플랫폼의 모든 기능과 함께 사용하십시오.
- 실시간 거래를 위해 모델을 조정하기 전에 타사 도구를 사용하지 않고 전략 테스터에서 과거 데이터에 대한 모델의 동작을 테스트할 수 있습니다.

MQL5는 ONNX 작업을 위해 다음과 같은 함수를 제공합니다.

함수	동작
OnnxCreate	*.onnx 파일에서 모델을 로드하여 ONNX 세션 생성
OnnxCreateFromBuffer	ONNX 세션 생성, 데이터 배열에서 모델 로드
OnnxRelease	ONNX 세션 종료
OnnxRun	ONNX 모델 실행
OnnxGetInputCount	ONNX 모델의 입력 수 가져오기
OnnxGetOutputCount	ONNX 모델의 출력 수 가져오기
OnnxGetInputName	인덱스로 모델의 입력 이름 가져오기
OnnxGetOutputName	인덱스로 모델의 출력 이름 가져오기
OnnxGetInputTypeInfo	모델에서 입력 유형에 대한 설명 가져오기
OnnxGetOutputTypeInfo	모델에서 출력 유형에 대한 설명 가져오기
OnnxSetInputShape	모델의 입력 데이터의 형태를 인덱스로 설정
OnnxSetOutputShape	모델의 출력 데이터의 형태를 인덱스로 설정

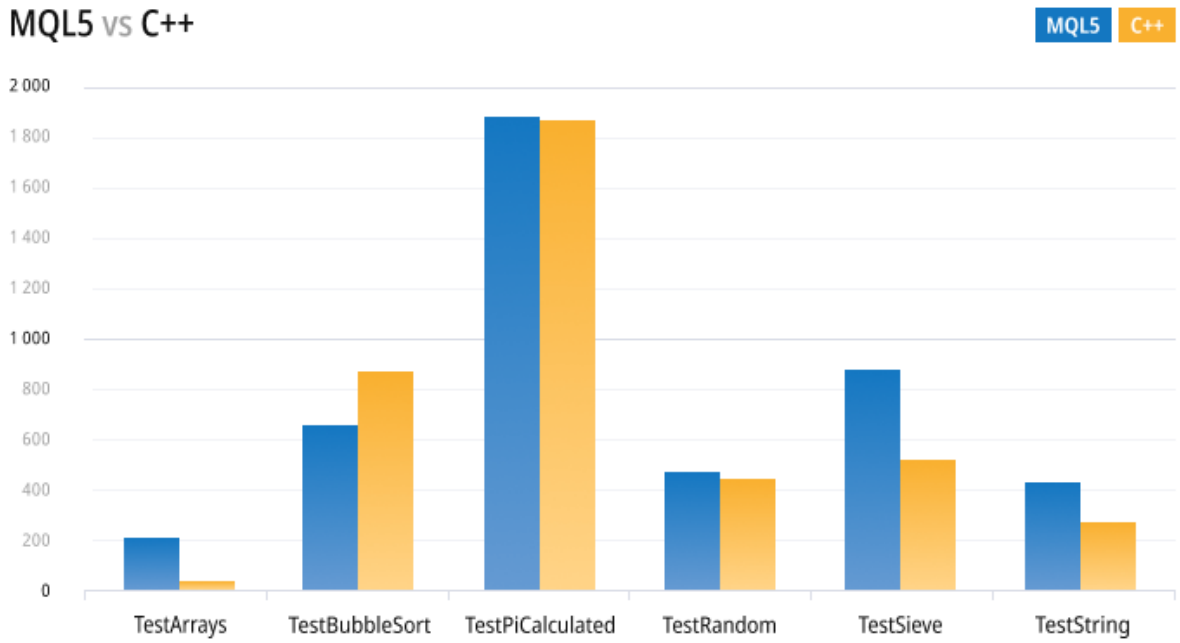
© 2000-2025, MetaQuotes Ltd.

MQL5에서 ONNX 지원

ONNX는 머신러닝 모델을 대표하는 개방형 형식의 빌드입니다. 이 ONNX 표준은 개발자가 다양한 프레임워크, 도구, 런타임 및 컴파일러와 함께 모델을 사용할 수 있도록 공통 연산자 세트 및 공통 파일 형식을 정의합니다.

따라서 개방형 ONNX 형식을 사용하면 서로 다른 플랫폼 및 기계 학습 툴킷 간에 머신러닝 모델을 수신하고 전송할 수 있습니다. ONNX는 MQL5 언어로 구현되어 AI 개발자가 MetaTrader 5 플랫폼에서 제공하는 고성능 실행 환경에서 생성된 모델을 실행할 수 있게합니다.

MQL5 실행 속도는 C++ 애플리케이션과 비슷합니다. 이는 MQL5 및 C++에서 실행한 표준 테스트의 결과로 입증됩니다. 바가 낮을수록 실행에 소요되는 시간(밀리초)이 줄어들고 결과가 더 좋습니다. 테스트는 Windows 10(빌드 17763) x64, Xeon E5-2630 v4 @2.20GHz, 메모리: 65457Mb에서 수행되었습니다.



새로운 비동기식 거래 작업 및 기본 ONNX 지원은 새로운 기회를 제공합니다. 이전에는 이러한 지원이 소수의 전문 AI 개발자 및 기관 거래자에게만 제공되었습니다. MQL5의 ONNX 지원을 통해 거래자는 선호하는 개발 환경에서 금융 시장 거래를 위한 모델을 교육한 다음 낮은 네트워크 비용, 빠른 주문 내역 업데이트 속도 및 비동기식 주문 접수등의 방식으로 거래할 수 있습니다.

현재 ONNX는 Microsoft, Facebook, Amazon 등과 같은 파트너 회사에서 개발 및 유지 관리되고 있으며 이는 이 개방형 프로젝트가 추가적으로 개발될 것임을 예상할수 있습니다.

© 2000-2025, MetaQuotes Ltd.

포맷 변환

ONNX는 다양한 머신러닝 도구 키트의 모델을 사용할 수 있게 하는 개방형 형식입니다. 이 형식은 다음을 포함한 많은 프레임워크에서 지원됩니다. Chainer,Caffee2 그리고 PyTorch.

모델을 ONNX 형식으로 변환하는 데 가장 널리 사용되는 도구 중 하나는 Microsoft의ONNXML도구입니다.

ONNXMLTools 설치 및 사용 지침은GitHub repo에서 찾을수 있습니다. 현재 지원되는 툴킷은 다음과 같습니다:

- Keras (keras2onnx converter의 랩퍼)
- Tensorflow (tf2onnx converter의 랩퍼)
- scikit-learn (skl2onnx converter의 랩퍼)
- Apple Core ML
- Spark ML (experimental)
- LightGBM
- libscm;
- XGBoost;
- H2O
- CatBoost

ONNXMLTools는 쉽게 설치할 수 있습니다. 설치 세부 사항 및 모델 변환 예는 프로젝트 페이지를 참조하십시오.https://github.com/onnx/onnxmltools#install.

© 2000-2025, MetaQuotes Ltd.

ONNX 모델 실행 시 입력 및 출력 값 자동 변환

MQL5의 현재 ONNX 버전은 input/output<에 대한 텐서만 지원합니다. 값. 텐서는 다음과 같은 데이터 유형의 요소가 있는 데이터 배열입니다:

ONNX type	Corresponds to MQL5 type
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_UINT8	uchar
ONNX_DATA_TYPE_INT8	char
ONNX_DATA_TYPE_UINT16	ushort
ONNX_DATA_TYPE_INT16	short
ONNX_DATA_TYPE_INT32	int
ONNX_DATA_TYPE_INT64	long
ONNX_DATA_TYPE_FLOAT16	—
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	uint
ONNX_DATA_TYPE_UINT64	ulong
ONNX_DATA_TYPE_COMPLEX64	—
ONNX_DATA_TYPE_COMPLEX128	complex
ONNX_DATA_TYPE_BFLOAT16	—
ONNX_DATA_TYPE_STRING	—

배열,벡터와 행렬만 (우리는 이것들을**Data**라고 부릅니다) 입력/출력 값으로 ONNX 모델에 공급될 수 있습니다.

매개변수 유형이 ONNX 모델의 매개변수 유형과 일치하지 않는 경우 OnnxRun은 ONNX_NO_CONVERSION 플래그를 지정 없이 호출됩니다. 자동 데이터 변환이 적용됩니다. 자동 변환은 ONNX 모델을 실행하기 전에 사용자 **Data**와 관련 변환과 함께 ONNX 텐서에 복사됩니다.

ONNX 모델이 자동 변환 없이 실행되면 모델은 추가적인 복사 없이**Data**를 사용하여 계산됩니다.

> **중요!** 자동 변환은 오버플로(자르기)를 제어하지 않으므로 데이터와 ONNX 모델에 입력되는 데이터 유형을 주의 깊게 모니터링해야 합니다.

자동 변환은 다음과 같은 ONNX 유형을 지원합니다.

- ONNX_DATA_TYPE_BOOL
- ONNX_DATA_TYPE_FLOAT
- ONNX_DATA_TYPE_UINT8

© 2000-2025, MetaQuotes Ltd.

- ONNX_DATA_TYPE_INT8
- ONNX_DATA_TYPE_UINT16
- ONNX_DATA_TYPE_INT16
- ONNX_DATA_TYPE_INT32
- ONNX_DATA_TYPE_INT64
- ONNX_DATA_TYPE_FLOAT16
- ONNX_DATA_TYPE_DOUBLE
- ONNX_DATA_TYPE_UINT32
- ONNX_DATA_TYPE_UINT64
- ONNX_DATA_TYPE_COMPLEX64
- ONNX_DATA_TYPE_COMPLEX128

지원되지 않는 유형:

- ONNX_DATA_TYPE_BFLOAT16
- ONNX_DATA_TYPE_STRING

Tensor 유형별 자동 변환 규칙

만약MQL5 유형이 모델에서 지원하는 유형 목록에 포함되지 않은 경우에 ONNX 모델을 실행하면 ERR_ONNX_NOT_SUPPORTED 오류(오류 코드 5802)이 반환됩니다.

참고: 자동 변환 중에는 색상 유형은 unit로 처리되지만 날짜 시간은 long으로 처리됩니다.

입력 값의 자동 변환

ONNX 유형(텐서 항목 유형)	자동 변환이 지원되는 MQL5 유형
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex 변환하는 동안**Data** 요소는 0에 대해 비교하여 확인됩니다.
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT16	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	See ONNX_DATA_TYPE_FLOAT

© 2000-2025, MetaQuotes Ltd.

ONNX 유형(텐서 항목 유형)	자동 변환이 지원되는 MQL5 유형
ONNX_DATA_TYPE_INT64	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

출력 값의 자동 변환

ONNX 유형(텐서 항목 유형)	자동 변환이 지원되는 MQL5 유형
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, folat, double, complex 텐서 요소가 0이면 Data 요소가 0으로 설정됩니다. 이외에는 값은 1입니다.
ONNX_DATA_TYPE_FLOAT16	float, double
ONNX_DATA_TYPE_FLOAT	char, uchar, short, ushort, int, color, uint, datetime, long, ulong, float, double
ONNX_DATA_TYPE_UINT8	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT8	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT16	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT16	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT32	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_INT64	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_DOUBLE	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT32	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_UINT64	See ONNX_DATA_TYPE_FLOAT
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

더 참조할 만한 내용

타입 캐스팅

© 2000-2025, MetaQuotes Ltd.

모델 생성

ONNX 형식으로 준비된 모델을 얻기 위해 여러 가지 메서드를 사용할 수 있습니다. 인기있는 라이브러리인 ONNX Model Zoo에는 다양한 유형의 작업에 대해 사전 훈련된 여러 ONNX 모델이 포함되어 있습니다. 이 컬렉션의 장점은 각 모델의 노트북이 교육 데이터 세트에 대한 링크와 모델 아키텍처를 설명하는 원본 문서에 대한 참조를 포함하고 있다는 것입니다.

대부분의 머신러닝 프레임워크는 Python을 사용합니다. Python용 ONNX 런타임을 설치하려면 다음 명령 중 하나를 사용합니다:

```
pip install onnxruntime        # CPU build
pip install onnxruntime-gpu    # GPU build
```

Python에서 ONNX 런타임을 호출하려면 다음 명령을 사용합니다.

```
import onnxruntime
session = onnxruntime.InferenceSession("path to model")
```

모델 입력 그리고 출력을 하려면 관련 모델의 문서를 확인하십시오. 다음과 같은 시각화 도구를 사용하여 모델을 볼 수도 있습니다. Netron 또는WinML dashboard. ONNX 런타임에서 모델의 메타데이터와 입력 및 출력을 쿼리할 수도 있습니다.

```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```

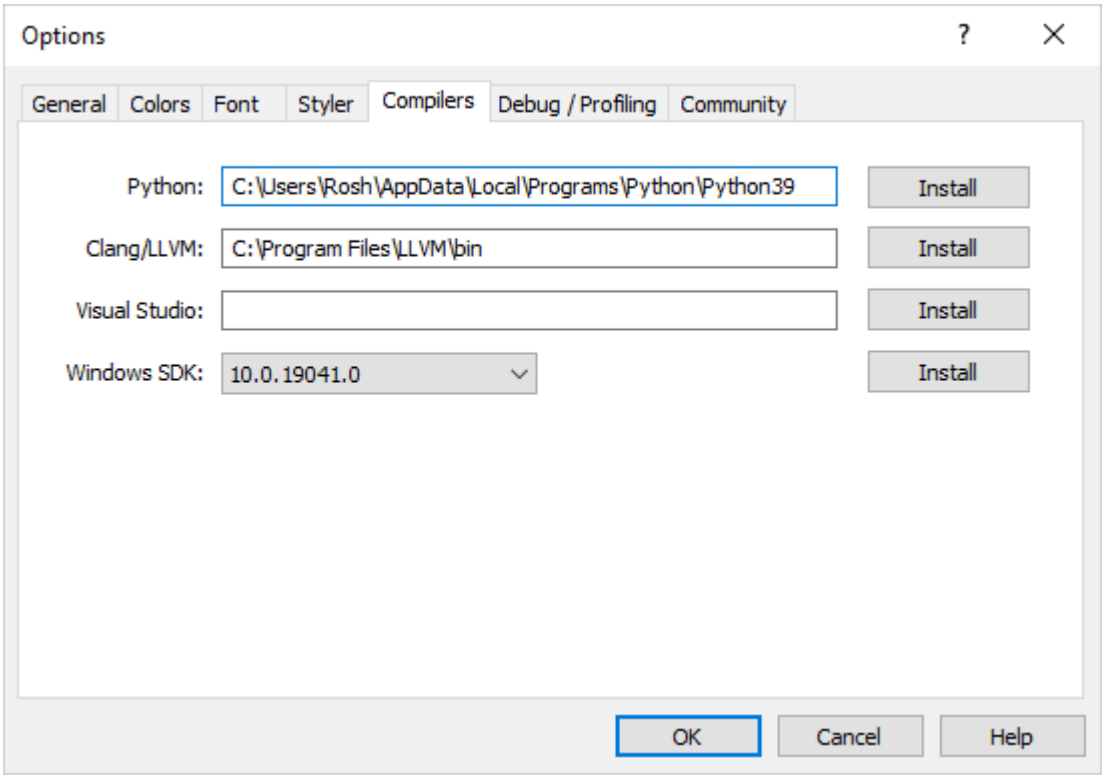
MetaTrader 5 터미널 또는 Python을 사용하는 MetaEditor에서 직접 ONNX 모델을 생성할 수 있습니다.

MetaTrader 5의 파이썬

MetaTrader 5는 Python 스크립트에 대해 틀에서 벗어난 지원을 제공합니다. 이러한 작업을 활성화하기 위해 터미널 개발자들이 Python용 MetaTrader5 모듈을 제공합니다 .https://pypi.org/project/MetaTrader5.

MetaEditor 통합 개발 환경에서는 MQL5에서 애플리케이션을 생성하는 것 외에도 편집기에서 직접 Python 스크립트를 실행할 수도 있습니다. 그렇게 하려면 MetaEditor의 설정에서 실행 파일의 경로를 지정하십시오.:

© 2000-2025, MetaQuotes Ltd.



컴퓨터에 Python이 설치되어 있지 않은 경우 설치를 클릭하여 설치 파일을 다운로드합니다.

MetaEditor에서 Python 스크립트를 생성하거나 터미널의 데이터 폴더에 업로드하고 F7(컴파일) 키를 사용하여 즉시 실행할 수 있습니다. 그러면 현재 차트에서 스크립트가 실행되는 MetaTrader 5 터미널이 열립니다. Python 콘솔(stdout, stderr)의 메시지는 Errors 섹션의 아래에 표시됩니다.

MetaTrader 5에서 모델 작업

MQL5 언어를 사용하면 MetaTrader 5 터미널에서 직접 ONNX 모델을 실행할 수 있습니다. 이는 세 단계로 수행됩니다.

1. Python과 같은 별도의 플랫폼에서 모델을 교육합니다.
2. 모델을 ONNX로 변환합니다.
3. ONNX 함수를 사용하여 ONNX 모델을 Expert Advisor에 포함합니다. 이후 MetaTrader 5 터미널에서 실행합니다.

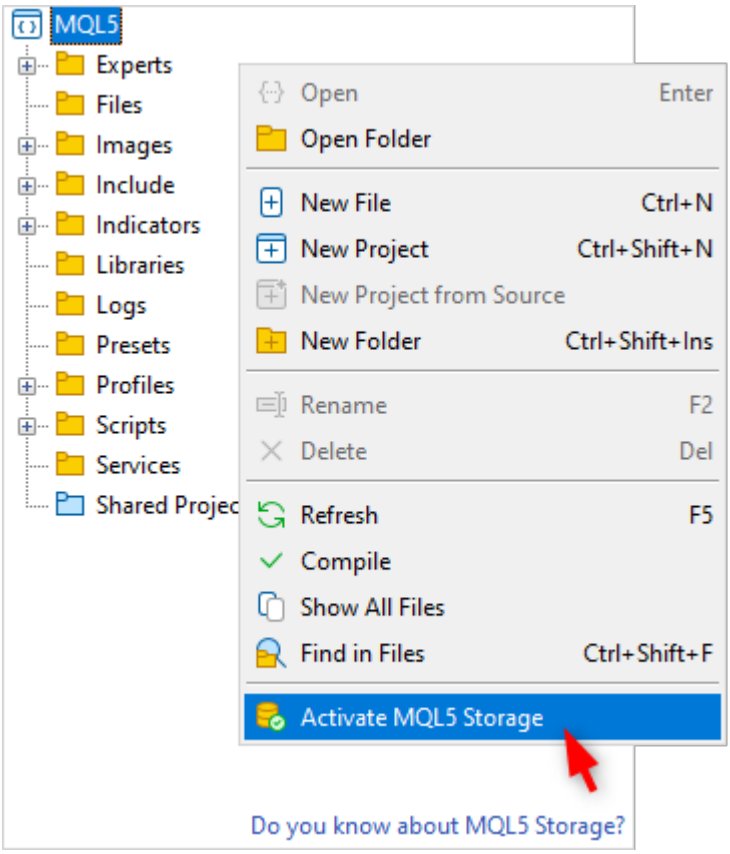
MQL5에서 파이썬과 통합함으로써 Python 스크립트를 실행하고 ONNX 모델을 MetaEditor에 저장하거나 MetaTrader 5의 차트에서 직접 실행할 수 있습니다. 터미널에서 필요할 때마다 미리 작성된 Python 스크립트를 사용하여 모델을 훈련할 수 있습니다. 라이브러리에는 ONNX 모델에 입력할 수 있는 가격 데이터를 얻기 위해 미리 만들어진 함수가 포함되어 있습니다.

- copy_rates_from – 지정된 날짜부터 바 가져오기
- copy_rates_from_pos – 지정된 인덱스에서 시작하는 바 가져오기
- copy_rates_range – 지정된 날짜 범위에 대한 바 가져오기
- copy_ticks_from – 지정된 날짜부터 틱을 가져옵니다.
- copy_ticks_range – 지정된 날짜 범위에 대한 틱 가져오기

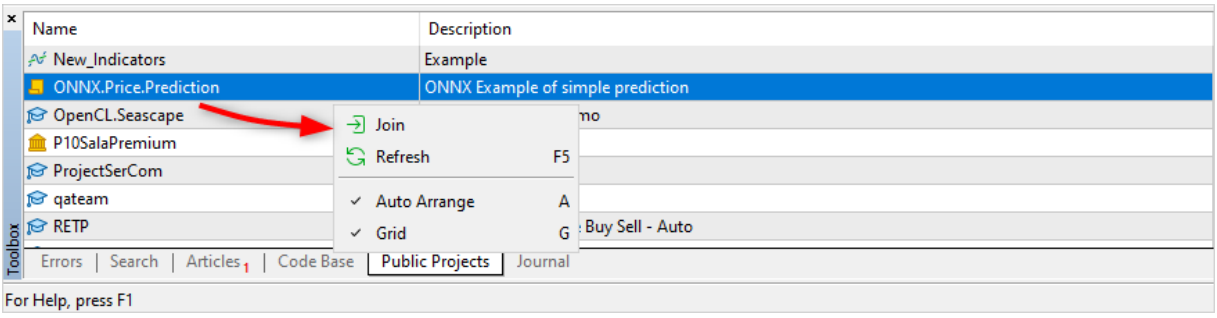
© 2000-2025, MetaQuotes Ltd.

모델 예

완성된 ONNX 모델의 예는 공개 프로젝트에서 사용할 수 있습니다.. MQL5 스토리지를 먼저 활성화해야 합니다. MetaEditor 설정에서 MQL5 로그인을 지정하여 Navigator에서(대소문자 구분).

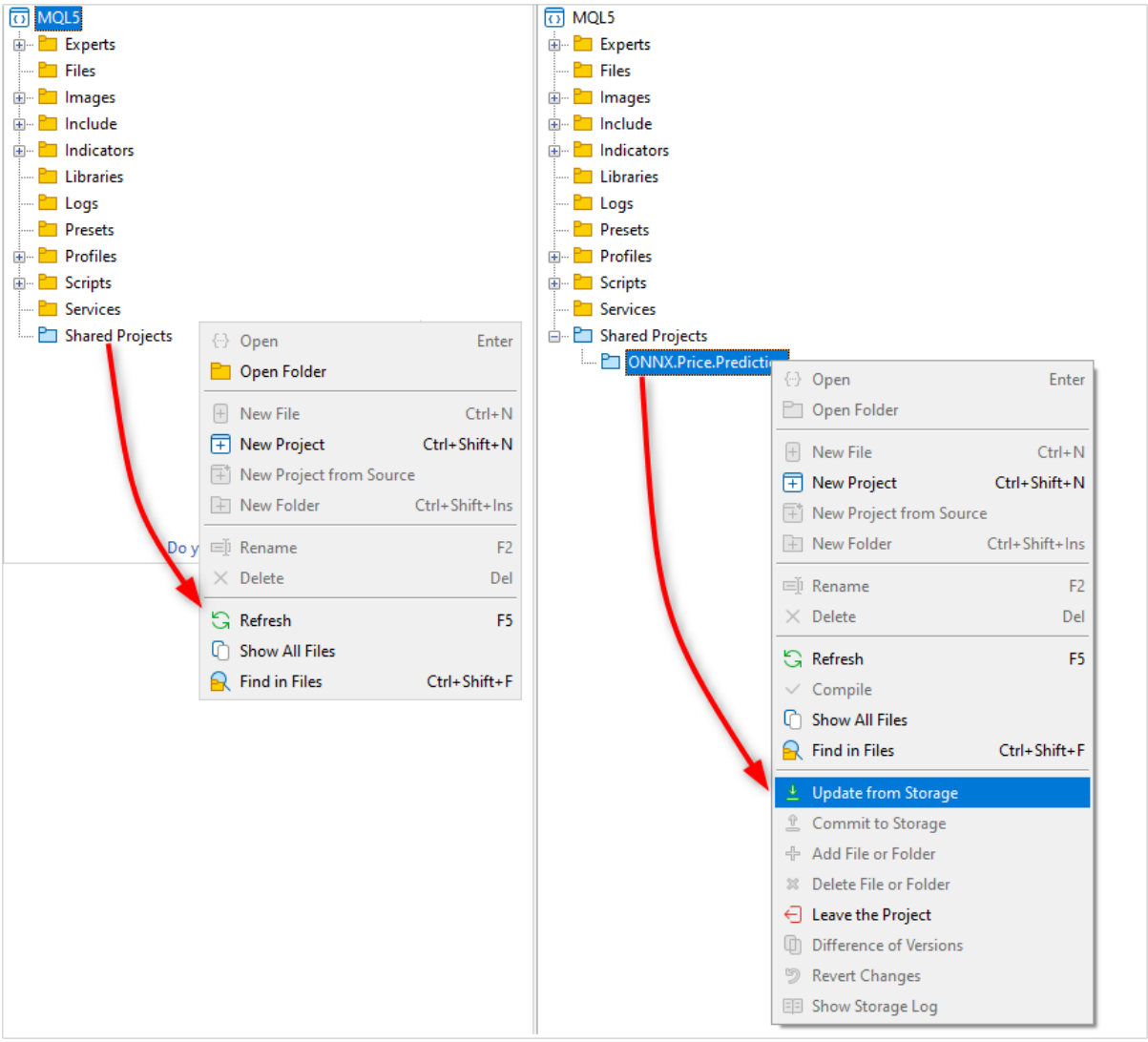


활성화 후 ONNX.Price.Prediction 프로젝트를 찾아 상황에 맞는 메뉴 명령을 통해 참여합니다.

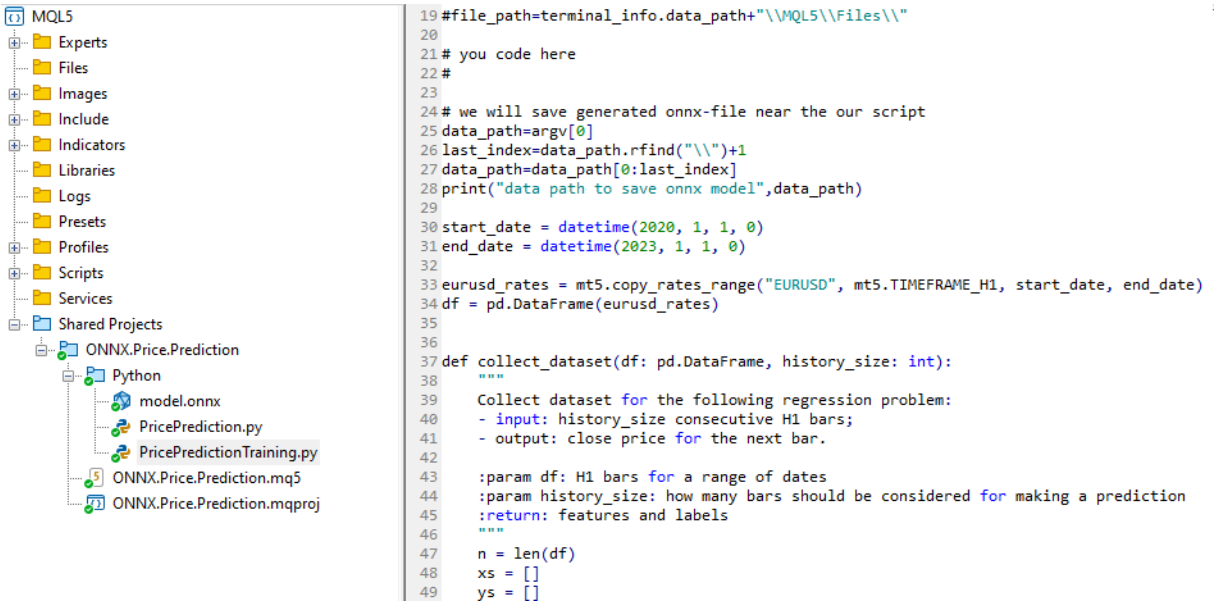


다음으로 MQL5 Storage에서 프로젝트를 업데이트합니다.

© 2000-2025, MetaQuotes Ltd.



프로젝트에는 ONNX 모델, 두 개의 Python 스크립트, 프로젝트 작업을 위한 MQL5 스크립트 및 MQL5 프로젝트 파일(ONNX.Price.Prediction.mqproj)이 포함되어 있습니다.

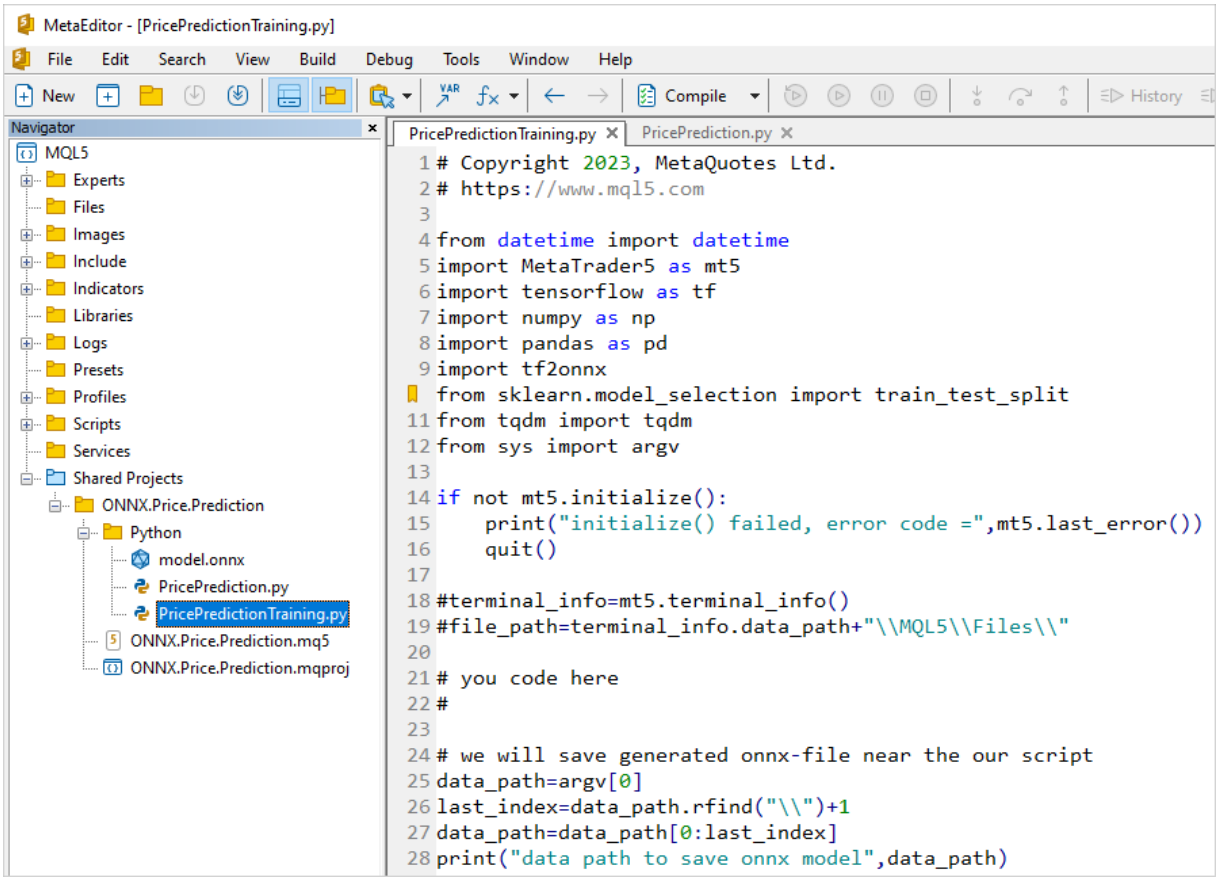


© 2000-2025, MetaQuotes Ltd.

프로젝트에 포함된 PricePredictionTraining.py 스크립트를 사용하여 ONNX 모델을 직접 만들 수 있습니다. 이렇게 하려면 먼저 명령줄에서 필요한 모듈을 설치해야 합니다.

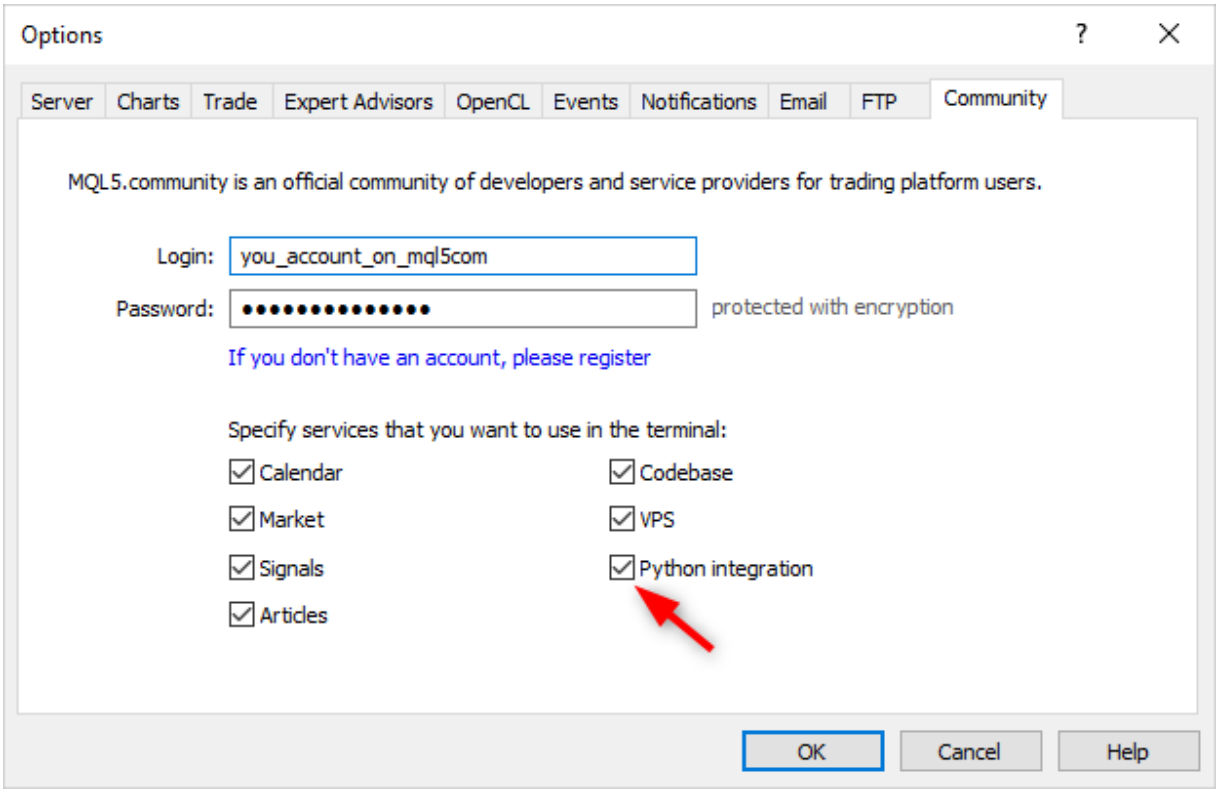
```
python.exe -m pip install --upgrade pip
python -m pip install --upgrade tensorflow
python -m pip install --upgrade pandas
python -m pip install --upgrade scikit-learn
python -m pip install --upgrade matplotlib
python -m pip install --upgrade tqdm
python -m pip install --upgrade metatrader5
python -m pip install --upgrade onnx==1.12
python -m pip install --upgrade tf2onnx
python -m pip install --upgrade numpy
python -m pip install onnxruntime
```

모듈을 설치한 후 MetaEditor에서 PricePredictionTraining.py 스크립트를 열고 컴파일 버튼 또는 F7 키를 사용하여 실행합니다.



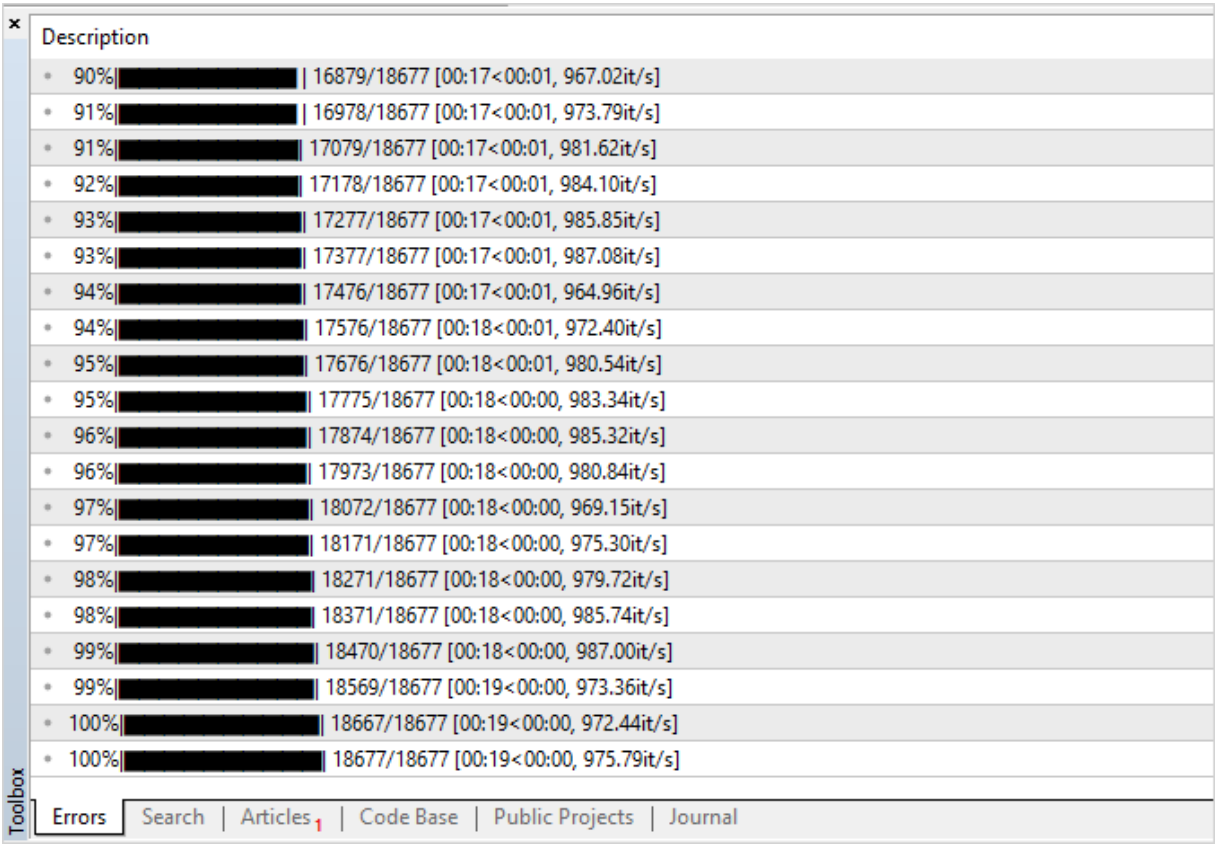
© 2000-2025, MetaQuotes Ltd.

Python 스크립트를 실행하기 전에 MetaTrader 5 터미널이 EURUSD 심볼이 있는 서버에 연결되어 있는지 확인하십시오. 예를 들어 MetaQuotes-Demo 서버에 연결하고 터미널 설정에서 "Integration with Python"을 확인합니다.



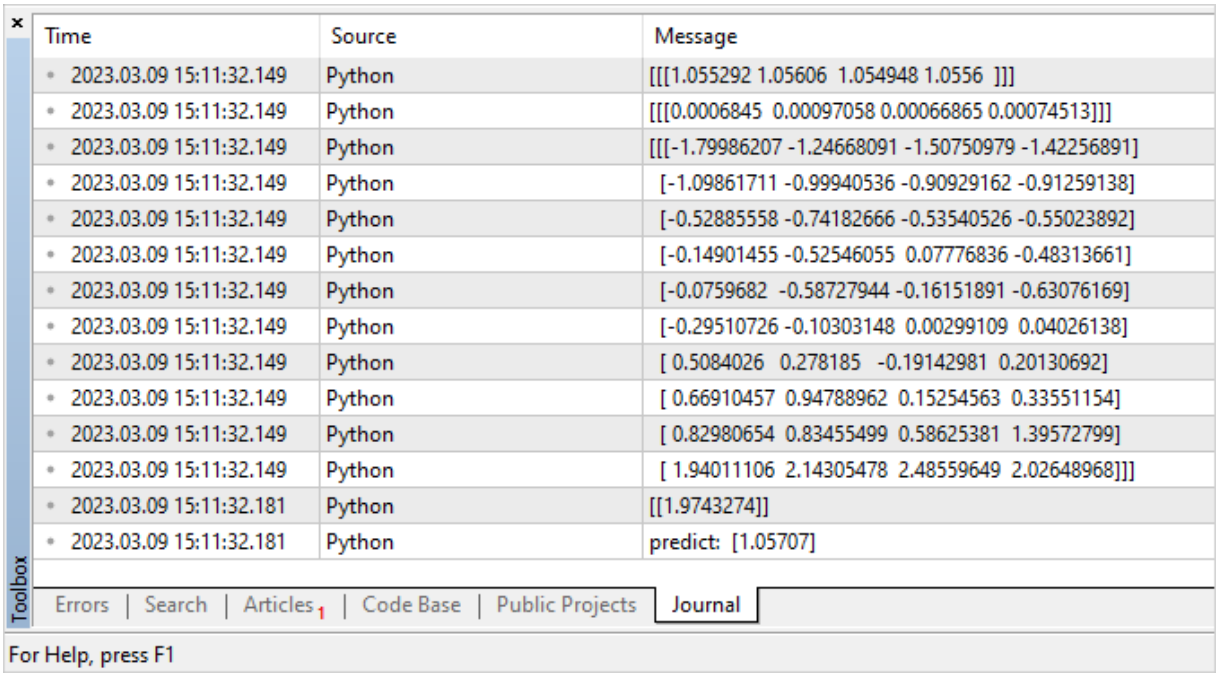
네트워크를 훈련하는 동안 MetaEditor는 훈련이 완료될 때까지 Python 스크립트에서 메시지를 인쇄합니다.

© 2000-2025, MetaQuotes Ltd.



결과가 100%이면 ONNX 모델이 준비되고 다음 위치의 프로젝트 폴더에 저장됩니다. <terminal data directory> \MQL5\Shared Projects\ONNX.Price.Prediction\Python.

F7을 눌러 두 번째 스크립트인 PricePrediction.py를 실행하여 결과 모델을 확인할 수 있습니다.



© 2000-2025, MetaQuotes Ltd.

모델 실행

MQL5에서 ONNX 모델을 실행하려면 다음과 같은 3단계를 거치세요:

1. 다음을 사용하여 *.onnx 파일에서 모델을 로드합니다.OnnxCreate 함수 또는 배열 OnnxCreateFromBuffer.
2. 다음을 사용하여 입력 및 출력 데이터 쉐이프 지정OnnxSetInputShape와OnnxSetOutputShape 함수.
3. OnnxRun함수를 사용하여 모델을 실행합니다. 관련 입력 및 출력 매개변수에 전달합니다.
4. 필요한 경우 다음을 사용하여 모델 작업을 종료할 수 있습니다.OnnxRelease 함수.

ONNX 모델을 생성할 때 https://github.com/microsoft/onnxruntime/blob/rel-1.14.0/docs/OperatorKernels.md에 설명된 현재 제한 사항 및 금지 사항을 고려해야 합니다.

이러한 제한 사항의 일부 예는 다음과 같습니다:

Operation	지원되는 데이터 유형
ReduceSum	tensor(double), tensor(float), tensor(int32), tensor(int64)
Mul	tensor(bfloat16), tensor(double), tensor(float), tensor(float16), tensor(int32), tensor(int64), tensor(uint32), tensor(uint64)

아래는 공개 프로젝트의 MQL5 코드 예제입니다.ONNX.Price.Prediction.

```
const long   ExtOutputShape[] = {1,1};    // 모델의 출력 쉐이프
const long   ExtInputShape [] = {1,10,4}; // 모델의 입력 쉐이프
#resource "Python/model.onnx" as uchar ExtModel[]// 자원으로서의 모델
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- 10개 바
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- OHLC 벡터 세트 입력
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- 정규화 행렬을 채움
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- 입력 데이터 정규화
```

© 2000-2025, MetaQuotes Ltd.

```
   x_norm-=mm;
   x_norm/=ms;
//--- 모델 생성
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- 입력 데이터의 쉐이프 지정
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 출력 데이터의 쉐이프 지정
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 정규화된 입력 데이터를 부동 소수점 유형으로 변환
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- 여기에서 모델의 출력 데이터 예를 들어 가격 예측을 얻습니다.
   vectorf y_norm(1);
//--- 모델 실행
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 모델의 출력 값을 로그에 출력
   Print(y_norm);
//--- 역변환을 수행하여 예상 가격을 얻습니다.
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- 작업 완료
   OnnxRelease(handle);
   return(0);
  }
```

스크립트 실행 예:

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
```

© 2000-2025, MetaQuotes Ltd.

```
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
[0.28188983]
predicted 1.0559258806393044
```

MetaTrader 5 터미널은 계산을 위한 최적의 executor를 선택했습니다 —ONNX Runtime Execution Provider. 이 예에서 모델은 CPU에서 실행되었습니다.

이전 10개 바의 값을 기반으로 괜찮은 레벨의 종가 예측의 백분율을 계산하도록 스크립트를 수정해 보겠습니다.

```
#resource "Python/model.onnx" as uchar ExtModel[]// 자원으로서의 모델

#define TESTS 10000  // 테스트 데이터세트의 수
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
int OnStart()
  {
//--- 모델 생성
   long session_handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
   if(session_handle==INVALID_HANDLE)
     {
      Print("Cannot create model. Error ",GetLastError());
      return(-1);
     }

//--- 입력 텐서 크기가 모델에 대해 정의되어 있지 않으므로 명시적으로 지정합니다.
//--- 첫 번째 인덱스는 배치 크기, 두 번째 인덱스는 시리즈 크기, 세 번째 인덱스는 시리즈 수(OHLC
   const long input_shape[]={1,10,4};
   if(!OnnxSetInputShape(session_handle,0,input_shape))
     {
      Print("OnnxSetInputShape error ",GetLastError());
      return(-2);
     }

//--- 출력 텐서 크기가 모델에 개해 정의되어 있지 않으므로 명시적으로 지정합니다.
//--- 첫 번째 인덱스는 배치 크기이며 입력 텐서의 배치 크기와 일치해야 합니다.
//--- 두 번째 인덱스는 예상 가격들의 수입니다(여기서는 종가만 예측됨).
   const long output_shape[]={1,1};
   if(!OnnxSetOutputShape(session_handle,0,output_shape))
     {
      Print("OnnxSetOutputShape error ",GetLastError());
      return(-3);
     }
//--- 테스트 실행
   vector closes(TESTS);      // 검증 가격을 저장할 벡터
```

© 2000-2025, MetaQuotes Ltd.

```
   vector predicts(TESTS);    // 구한 예측들을 저장할 벡터
   vector prev_closes(TESTS); // 이어지는 가격들을 저장할 벡터

   matrix rates;              // OHLC 시리즈를 얻기 위한 행렬
matrixsplitted[2];// 시리즈를 테스트와 검증으로 나누는 두 부분 행렬
   ulong  parts[]={10,1};     // 나누어진 부분 행렬의 크기

//--- 이전 바부터 시작
   for(int i=1; i<=TESTS; i++)
     {
      //--- 11개의 바를 얻음
      rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,i,11);
//--- 행렬을 테스트와 검증으로 나눕니다.
      rates.Vsplit(parts,splitted);
//--- 유효성 검증 매트릭스에서 종가를 가져옵니다.
      closes[i-1]=splitted[1][3][0];
//--- 테스트된 시리즈의 마지막 종가
      prev_closes[i-1]=splitted[0][3][9];

//--- 10개의 바의 테스트 매트릭스를 테스팅에 제출
      predicts[i-1]=PricePredictionTest(session_handle,splitted[0]);
//--- 런타임 에러
      if(predicts[i-1]<=0)
        {
         OnnxRelease(session_handle);
         return(-4);
        }
     }
//--- 작업 완료
   OnnxRelease(session_handle);
//--- 가격 움직임이 올바르게 예측되었는지를 평가
   int    right_directions=0;
   vector delta_predicts=prev_closes-predicts;
   vector delta_actuals=prev_closes-closes;

   for(int i=0; i<TESTS; i++)
      if((delta_predicts[i]>0 && delta_actuals[i]>0) || (delta_predicts[i]<0 && delta_
         right_directions++;
   PrintFormat("right direction predictions = %.2f%%",(right_directions*100.0)/double
//---
   return(0);
  }
/+------------------------------------------------------------------+
//| 데이터 준비하고 모델 실행                                        |
/+------------------------------------------------------------------+
double PricePredictionTest(const long session_handle,matrix& rates)
  {
   static matrixf input_data(10,4); // 변환된 입력을 위한 행렬
   static vectorf output_data(1);   // 결과를 받을 벡터
```

© 2000-2025, MetaQuotes Ltd.

```
   static matrix mm(10,4);           //수평 벡터의 행렬 Mean
   static matrix ms(10,4);           // 수평 벡터의 Std

//--- OHLC 수직 벡터 세트가 모델에 입력되어야 합니다.
   matrix x_norm=rates.Transpose();
//--- 가격 정규화
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
   x_norm-=mm;
   x_norm/=ms;

//--- 모델 실행
   input_data.Assign(x_norm);
   if(!OnnxRun(session_handle,ONNX_DEBUG_LOGS,input_data,output_data))
     {
      Print("OnnxRun error ",GetLastError());
      return(0);
     }
//--- 출력 값에서 가격을 비정규화
   double y_pred=output_data[0]*s[3]+m[3];

   return(y_pred);
  }
```

스크립트 실행: 예측 정확도는 약 51%입니다.

```
ONNX: Creating and using per session threadpools since use_per_session_threads_ is tru
ONNX: Dynamic block base set to 0
ONNX: Initializing session.
ONNX: Adding default CPU execution provider.
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Total shared scalar initializer count: 0
ONNX: Total fused reshape node count: 0
ONNX: Use DeviceBasedPartition as default
ONNX: Saving initialized tensors.
ONNX: Done saving initialized tensors
ONNX: Session successfully initialized.
올바른 방향 예측 = 51.34%
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

전략 테스터에서 모델 검증

금융 시장을 위해 생성된 모델은 MetaTrader 5 터미널의 전략 테스터에서 검증할 수 있습니다. 이것은 시장 환경과 거래 조건을 추가로 만들어 사용할 필요가 없는 가장 빠르고 편리한 옵션입니다.

모델을 테스트하기 위해 공개 프로젝트 ONNX.Price.Prediction의 코드를 기반으로 Expert Advisor를 생성해 보겠습니다. 이를 위해서는 약간의 수정이 필요합니다.

모델 생성을OnInit 함수로 이동합니다. onnx 세션이 OnDeinit에서 종료됩니다. 기본 모델 작업 블록을 OnTick 핸들러에 놓습니다.

또한 실제의 종가와 예측을 비교하는 데 필요한 이전 두 바의 종가의 획득을 추가합니다.

Expert Advisor 코드는 작고 읽기 쉽습니다.

```
const long   ExtInputShape [] = {1,10,4}; // 모델의 입력 쉐이프
const long   ExtOutputShape[] = {1,1};    // 모델의 출력 쉐이프
#resource "Python/model.onnx" as uchar ExtModel[];// 자원으로서의 모델

long handle;          // 모델 핸들
ulong predictions=0; // 예측 카운터
ulong confirmed=0;   // 예측 카운터 성공
/+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
/+------------------------------------------------------------------+
int OnInit()
  {
//--- 기본 확인
   if(_Symbol!="EURUSD")
     {
      Print("Symbol must be EURUSD, testing aborted");
      return(-1);
     }
   if(_Period!=PERIOD_H1)
     {
      Print("Timeframe must be H1, testing aborted");
      return(-1);
     }
//--- 모델 생성
   handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- 입력 데이터의 쉐이프 지정
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 출력 데이터의 쉐이프 지정
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
```

© 2000-2025, MetaQuotes Ltd.

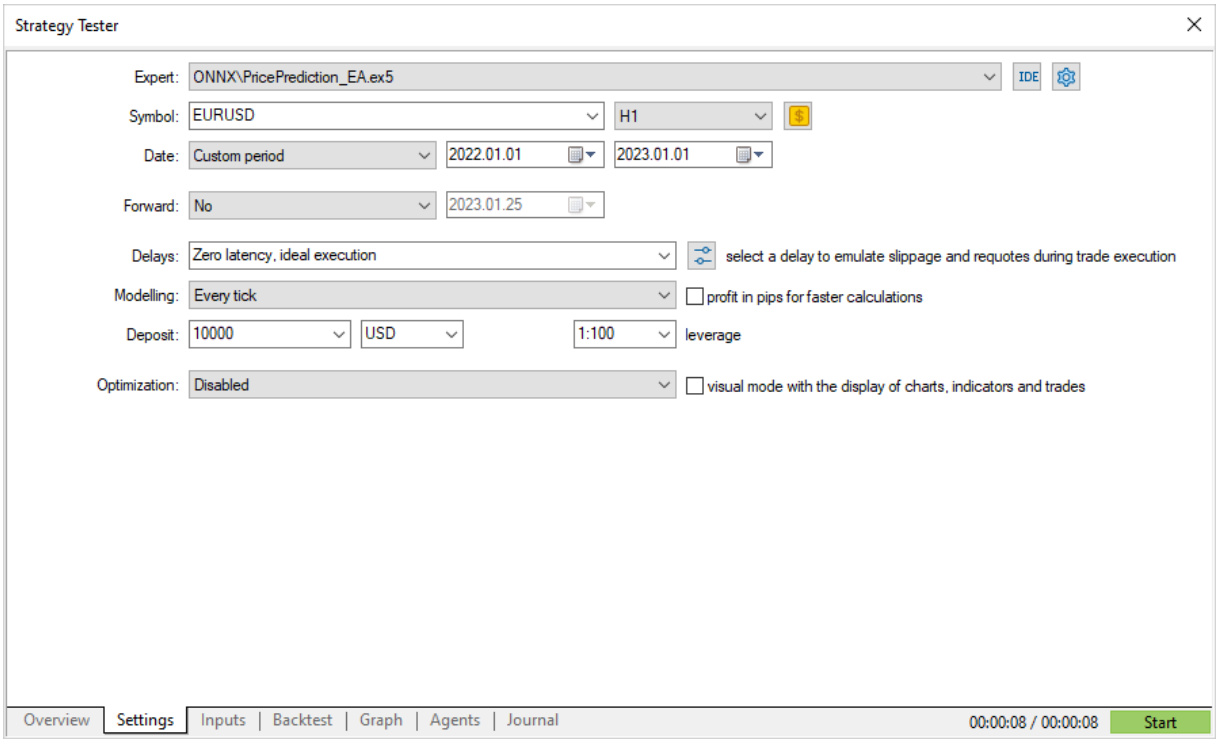
```
      {
       Print("OnnxSetOutputShape failed, error ",GetLastError());
       OnnxRelease(handle);
       return(-1);
      }
//---
   return(INIT_SUCCEEDED);
  }
/+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
/+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 모델 작업 완료
   OnnxRelease(handle);
//--- 예측 통계 계산 및 출력
   PrintFormat("Successfull predictions = %.2f %%",confirmed*100./double(predictions));
  }
/+------------------------------------------------------------------+
//| Expert 틱 함수                                                   |
/+------------------------------------------------------------------+
void OnTick()
  {
   static datetime open_time=0;
   static double predict;
//--- 현재 바 오픈 시간 확인
   datetime time=iTime(_Symbol,_Period,0);
   if(time==0)
     {
      PrintFormat("Failed to get Time(0), error %d", GetLastError());
      return;
     }
//--- 개장 시간이 변경되지 않은 경우 다음 OnTick 호출까지 종료
   if(time==open_time)
      return;
//--- 마지막 완성된 바 2개의 종가를 가져옵니다.
   double close[];
   int recieved=CopyClose(_Symbol,_Period,1,2,close);
   if(recieved!=2)
     {
      PrintFormat("CopyClose(2 bars) failed, error %d",GetLastError());
      return;
     }
   double delta_predict=predict-close[0]; // 예측한 가격 변경
   double delta_actual=close[1]-close[0]; // 실제 가격 변경
   if((delta_predict>0 && delta_actual>0) || (delta_predict<0 && delta_actual<0))
      confirmed++;

//--- 새개 바의 종가를 계산하여 다음 바의 가격을 확인합니다.
```

© 2000-2025, MetaQuotes Ltd.

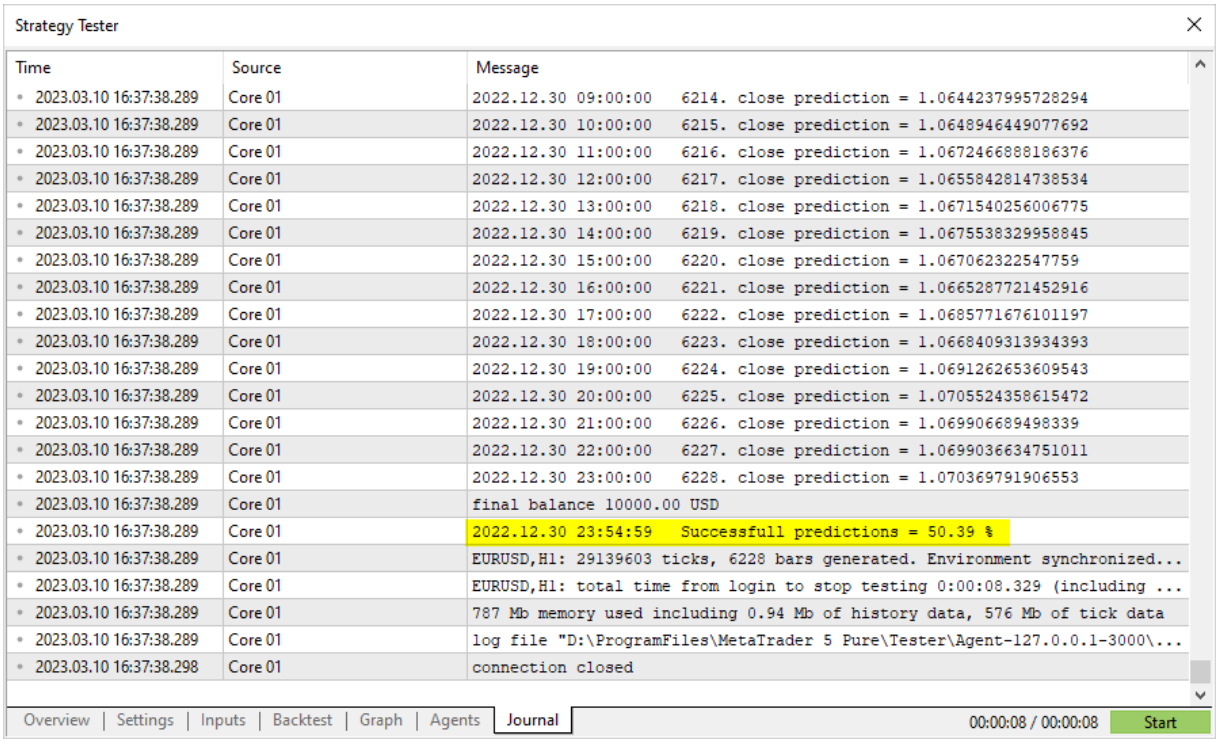
```
   matrix rates;
//--- 10개 바
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,1,10))
      return;
//--- OHLC 벡터 세트 입력
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- 정규화 행렬을 채움
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- 입력 데이터 정규화
   x_norm-=mm;
   x_norm/=ms;
//--- 정규화된 입력 데이터를 부동 소수점 유형으로 변환
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- 여기에서 모델의 출력 데이터 예를 들어 가격 예측을 얻습니다.
   vectorf y_norm(1);
//--- 모델 실행
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
     }
//--- 역변환을 수행하여 예상 가격을 얻고 새로운 바에서 유효성을 검사합니다.
   predict=y_norm[0]*s[3]+m[3];
   predictions++;  // 예측 카운터 증대
   Print(predictions,". close prediction = ",predict);
//--- 다음 틱을 확인하기 위해 바 오프닝 시간을 저장
   open_time=time;
  }
```

Expert Advisor를 컴파일하고 2022년도로 테스트를 실행합니다. 모델이 훈련되는 데이터인 H1 기간으로 EURUSD를 지정하십시오. 틱 모델링 모드는 코드가 새로운 바의 생성을 확인하기 때문에 무시할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.



테스팅 저널을 실행하고 결과를 확인하십시오. 결과에 따르면 2022년 예측의 50% 이상이 정확했음을 보여줍니다.



예비 모델 테스트가 만족스러운 결과를 생성했다면 이 모델을 기반으로 본격적인 거래 전략을 작성하기 시작할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

OnnxCreate

*.onnx 파일에서 모델을 로드하여 ONNX 세션을 만듭니다.

```
long  OnnxCreate(
   string  filename,  // 파일 경로
   uint    flags      // 모델을 생성할 플래그
   );
```

패러미터

filename

【n】₩MQL5₩Files₩폴더에 대한 모델의 *.onnx 파일 경로입니다.

flags

【n】ENUM_ONNX_FLAGS 플래그 모델 생성 모드 설명: ONNX_COMMON_FOLDER 및 ONNX_DEBUG_LOGS.

반환 값

생성된 세션의 핸들 또는INVALID_HANDLE은 오류가 발생할 경우. error 코드 호출을 위한 GetLastError 함수.

주의

지정된 파일이 디스크에서 발견되지 않으면 시스템은 이름에 '.onnx' 확장자를 추가하고 파일 열기를 다시 시도합니다.

© 2000-2025, MetaQuotes Ltd.

OnnxCreateFromBuffer

데이터 배열에서 모델을 로드하여 ONNX 세션을 만듭니다.

```
long  OnnxCreateFromBuffer(
   const uchar&  buffer[],   // 배열 참조
   ulong         flags       // 모델 생성 플래그
   );
```

패러미터

buffer

[in] ONNX 모델 데이터가 있는 배열

flags

【n】ENUM_ONNX_FLAGS 플래그 모델 생성 모드 설명: ONNX_COMMON_FOLDER 및 ONNX_DEBUG_LOGS.

반환 값

생성된 세션의 핸들 또는INVALID_HANDLE은 오류가 발생할 경우. error 코드 호출을 위한 GetLastError 함수.

© 2000-2025, MetaQuotes Ltd.

OnnxRelease

ONNX 세션을 종료

```
bool  OnnxRelease(
   long   onnx_handle  // ONNX 세션 핸들
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

반환 값

성공하면 true를 반환합니다. 그렇지 않으면 거짓을 반환합니다. error 코드를 호출하기 위한 GetLastError 함수.

© 2000-2025, MetaQuotes Ltd.

OnnxRun

ONNX 모델을 실행.

```
bool  OnnxRun(
   long    onnx_handle,  // ONNX 세션 핸들
   ulong   flags,        // 실행 모드를 설명하는 플래그
   ...                   // 모델의 입력과 출력
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

flags

【n】 ENUM_ONNX_FLAGS의 플래그로 실행 모드를 설명: ONNX_DEBUG_LOGS 및 ONNX_NO_CONVERSION.

...

[in] [out] 모델 입력과 출력.

성공하면 true를 반환하고 그렇지 않으면 false를 반환. error 코드를 얻기 위해서는 GetLastError 함수를 호출합니다.

ENUM_ONNX_FLAGS

ID	설명
ONNX_DEBUG_LOGS	디버그 로그 출력
ONNX_NO_CONVERSION	자동 변환을 비활성화하고 사용자 데이터를 그대로 사용
ONNX_COMMON_FOLDER	Common₩files 폴더에서 모델 파일을 로드합니다; 값은FILE_COMMON flag와 같습니다.

예:

```
const long                              ExtOutputShape[] = {1,1};     // 모델 출력 쉐이프
const long                              ExtInputShape [] = {1,10,4}; // 모델 입력 폼
#resource "Python/model.onnx" as uchar ExtModel[]                    // 자원으로서의 모델
/+------------------------------------------------------------------+
//| 프로그램 시작 함수 스크립트                                      |
/+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- 10개 바
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(-1);
//--- OHLC 벡터 세트 입력
   matrix x_norm=rates.Transpose();
   vector m=x_norm.Mean(0);
   vector s=x_norm.Std(0);
   matrix mm(10,4);
   matrix ms(10,4);
//--- 정규화 행렬을 채움
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- 입력 데이터 정규화
   x_norm-=mm;
   x_norm/=ms;
//--- 모델 생성
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- 입력 데이터의 쉐이프 지정
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 출력 데이터의 쉐이프 지정
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 정규화된 입력 데이터를 부동 소수점 유형으로 변환
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- 여기에서 모델의 출력 데이터 예를 들어 가격 예측을 얻습니다.
   vectorf y_norm(1);
//--- 모델 실행
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 모델의 출력 값을 로그에 출력
   Print(y_norm);
//--- 역변환을 수행하여 예상 가격을 얻습니다.
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 완료된 작업
   OnnxRelease(handle);
   return(0);
  };
```

더 참조할 만한 내용

OnnxSetInputShape, OnnxSetOutputShape

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputCount

ONNX 모델의 입력 수를 가져옵니다.

```
long  OnnxGetInputCount(
   long   onnx_handle  // ONNX 세션 핸들
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

반환 값

성공 시 입력 패러미터의 수를 반환합니다. 그렇지 않으면 -1을 반환합니다. error 코드를 호출하기 위한GetLastError 함수.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputCount

ONNX 모델의 출력 수를 가져옵니다.

```
long  OnnxGetOutputCount(
   long   onnx_handle  // ONNX 세션 핸들
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

반환 값

성공 시 출력 매개변수의 수를 반환합니다. 그렇지 않으면 -1을 반환합니다. error 코드를 호출하기 위한GetLastError 함수.

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputName

인덱스로 모델의 입력 이름을 가져옵니다.

```
string  OnnxGetInputName(
   long   onnx_handle,  // ONNX 세션 핸들
   long   index         // 패러미터 인덱스
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

index

【n】0부터 시작하는 입력 매개변수의 인덱스입니다.

반환 값

성공 시 입력 매개변수의 이름을 반환합니다. 그렇지 않으면 NULL을 반환합니다. error 코드를 호출하기 위한GetLastError 함수.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputName

모델의 출력 이름을 인덱스로 가져옵니다.

```
string  OnnxGetOutputName(
   long   onnx_handle,  // ONNX 세션 핸들
   long   index         // 패러미터 인덱스
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

index

【n】0부터 시작하는 출력 매개변수의 인덱스입니다.

반환 값

성공 시 출력 매개변수의 이름을 반환합니다. 그렇지 않으면 NULL을 반환합니다. error 코드를 호출하기 위한GetLastError 함수.

© 2000-2025, MetaQuotes Ltd.

OnnxGetInputTypeInfo

모델에서 입력 유형에 대한 설명을 가져옵니다.

```
bool  OnnxGetInputTypeInfo(
   long           onnx_handle,  // ONNX 세션 핸들
   long           index,        //패러미터 인덱스
   OnnxTypeInfo&  typeinfo      // 패러미터 타입 설명
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

index

【n】 0부터 시작하는 입력 매개변수의 인덱스입니다.

typeinfo

[out]OnnxTypeInfo 입력 매개변수의 유형을 설명하는 구조입니다.

반환 값

성공하면 true를 반환합니다. 그렇지 않으면 거짓을 반환합니다. error 코드를 호출하기 위한 GetLastError 함수.

© 2000-2025, MetaQuotes Ltd.

OnnxGetOutputTypeInfo

모델에서 출력 유형에 대한 설명을 가져옵니다.

```
bool  OnnxGetOutputTypeInfo(
   long           onnx_handle,  // ONNX 세션 핸들
   long           index,        //패러미터 인덱스
   OnnxTypeInfo&  typeinfo      // 패러미터 타입 설명
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

index

【n】 0부터 시작하는 출력 매개변수의 인덱스입니다.

typeinfo

[out]OnnxTypeInfo는 출력 매개변수의 유형을 설명하는 구조입니다.

반환 값

성공하면 true를 반환합니다. 그렇지 않으면 거짓을 반환합니다. error 코드를 호출하기 위한 GetLastError 함수.

© 2000-2025, MetaQuotes Ltd.

OnnxSetInputShape

모델의 입력 데이터의 형태를 인덱스별로 설정합니다.

```
bool  OnnxSetInputShape(
   long           onnx_handle,  // ONNX 세션 핸들
   long           input_index,  // 입력 패러미터 인덱스
   const ulong&   shape[]       // 입력 데이터 쉐이프를 설명하는 배열
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

input_index

【n】 0부터 시작하는 입력 매개변수의 인덱스입니다.

shape

[in] 모델의 입력 데이터의 쉐이프를 설명하는 배열입니다.

반환 값

성공 시 입력 매개변수의 이름을 반환합니다. 그렇지 않으면 NULL을 반환합니다. error 코드를 호출하기 위한GetLastError 함수.

예:

```
//---- 모델의 입력 및 출력 데이터의 쉐이프를 설명합니다.
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- 모델 생성
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- 입력 데이터의 쉐이프 지정
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 출력 데이터의 쉐이프 지정
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

더 참조할 만한 내용

OnnxSetOutputShape

© 2000-2025, MetaQuotes Ltd.

OnnxSetOutputShape

모델의 출력 데이터의 형태를 인덱스별로 설정합니다.

```
bool  OnnxSetOutputShape(
   long           onnx_handle,   // ONNX 세션 핸들
   long           output_index,  // 출력 패러미터 인덱스
   const ulong&   shape[]        // 출력 데이터 모양을 설명하는 배열
   );
```

패러미터

onnx_handle

[in] 다음을 통해 생성된 ONNX session 객체 핸들 OnnxCreate or OnnxCreateFromBuffer.

output_index

【n】 0부터 시작하는 출력 매개변수의 인덱스입니다.

shape

[in] 모델의 출력 데이터 모양을 설명하는 배열입니다.

반환 값

성공 시 입력 매개변수의 이름을 반환합니다. 그렇지 않으면 NULL을 반환합니다. error 코드를 호출하기 위한GetLastError 함수.

예:

```
//---- 모델의 입력 및 출력 데이터의 쉐이프를 설명합니다.
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- 모델 생성
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- 입력 데이터의 쉐이프 지정
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- 출력 데이터의 쉐이프 지정
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

더 참조할 만한 내용

OnnxSetInputShape

© 2000-2025, MetaQuotes Ltd.

데이터 구조

다음 데이터 구조는 ONNX 모델 작업에 사용됩니다:

OnnxTypeInfo

구조는 ONNX 모델의 input또는output parameter의 유형을 설명합니다.

```
struct OnnxTypeInfo
 {
  ENUM_ONNX_TYPE         type;           // 매개변수 타입
  OnnxTensorTypeInfo     tensor;         // 텐서 설명
  OnnxMapTypeInfo       map;             // 지도 설명
  OnnxSequenceTypeInfo   sequence;       // 시퀀스 설명
 };
```

텐서(ONNX_TYPE_TENSOR)만 input으로 사용할 수 있습니다. 이 경우 OnnxTypeInfo::tensor 필드만 값들로 채워지고 다른 필드(map 및 sequence)는 정의되지 않습니다.

세 가지 OnnxTypeInfo 유형(ONNX_TYPE_TENSOR, ONNX_TYPE_MAP 또는 ONNX_TYPE_SEQUENCE) 중 하나만 input으로 사용할 수 있습니다. 해당 하위 구조(OnnxTypeInfo::tensor, OnnxTypeInfo::map 또는 OnnxTypeInfo::sequence)는 유형에 따라 채워집니다.

OnnxTensorTypeInfo

구조는 다음의 텐서를 설명합니다. ONNX 모델의 input또는output parameter

```
struct OnnxTensorTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE  data_type;      // 텐서에서 데이터 타입
  const long                 dimensions[];   // 텐서에서 요소의 수
 };
```

OnnxMapTypeInfo

구조는 ONNX 모델의 출력 매개변수에서 얻은 지도를 설명합니다.

```
struct OnnxMapTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE  key_type;       // 키 타입
  const OnnxTypeInfo&        value_type;     // 값 타입
 };
```

OnnxSequenceTypeInfo

구조는 ONNX 모델의 출력 매개변수에서 얻은 시퀀스를 설명합니다.

```
struct OnnxSequenceTypeInfo
 {
```

© 2000-2025, MetaQuotes Ltd.

```
  const OnnxTypeInfo &        value_type;       //시퀀스에서 데이터 타입
};
```

ENUM_ONNX_TYPE

ENUM_ONNX_TYPE 열거형은 모델 매개변수의 유형에 대해 설명합니다.

ID	설명
ONNX_TYPE_UNKNOWN	Unknown
ONNX_TYPE_TENSOR	Tensor
ONNX_TYPE_SEQUENCE	Sequence
ONNX_TYPE_MAP	Map
ONNX_TYPE_OPAQUE	Abstract (opaque)
ONNX_TYPE_SPARSETENSOR	Sparse tensor

ENUM_ONNX_DATA_TYPE

ENUM_ONNX_DATA_TYPE 열거형은 사용된 데이터 유형에 대해 설명합니다.

ID	설명
ONNX_DATA_TYPE_UNDEFINED	Undefined
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_INT8	8-bit int
ONNX_DATA_TYPE_UINT16	16-bit uint
ONNX_DATA_TYPE_INT16	16-bit int
ONNX_DATA_TYPE_INT32	32-bit int
ONNX_DATA_TYPE_INT64	64-bit int
ONNX_DATA_TYPE_STRING	string
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT16	16-bit float
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	32-bit uint
ONNX_DATA_TYPE_UINT64	64-bit uint
ONNX_DATA_TYPE_COMPLEX64	64-bit complex number

© 2000-2025, MetaQuotes Ltd.

ID	설명
ONNX_DATA_TYPE_COMPLEX128	128-bit complex number
ONNX_DATA_TYPE_BFLOAT16	16-bit bfloat (Brain Floating Point)

ENUM_ONNX_FLAGS

ENUM_ONNX_FLAGS 열거는 모델 실행 모드에 대해 설명합니다.

ID	설명
ONNX_DEBUG_LOGS	디버그 로그 출력
ONNX_NO_CONVERSION	자동 변환을 비활성화하고 사용자 데이터를 그대로 사용
ONNX_COMMON_FOLDER	CommonWFiles 폴더에서 모델 파일을 로드합니다; 값은FILE_COMMON flag와 같습니다.

ONNX 모델 작업할 때 배열 변환

머신 러닝 작업에 항상 더 높은 계산 정확도가 필요한 것은 아닙니다. 계산 속도를 높이기 위해 일부 모델은 Float16 및 심지어 Float8과 같은 정밀도가 낮은 데이터 유형을 사용하기도 합니다. 사용자가 관련 데이터를 모델에 입력할 수 있도록 MQL5는 표준 MQL5 유형을 특수 FP16 및 FP8 유형으로 변환하는 네 가지 특수 함수를 제공합니다.

함수	액션
ArrayToFP16	float 또는 double 유형의 배열을 주어진 형식의 ushort 유형의 배열에 복사합니다
ArrayToFP8	float 또는 double 유형의 배열을 지정된 형식의 uchar 유형의 배열에 복사합니다
ArrayFromFP16	ushort 유형의 배열을 주어진 형식의 float 또는 double 유형의 배열로 복사합니다
ArrayFromFP8	uchar 유형의 배열을 지정된 형식의 float 또는 double 유형의 배열로 복사합니다

이러한 배열 변환 함수는 아래 열거에 지정된 특수 형식을 사용합니다.

ENUM_FLOAT16_FORMAT

ENUM_FLOAT16_FORMAT 열거형은 두 가지 FP16 유형 형식을 설명합니다.

ID	설명
FLOAT_FP16	표준 16비트 형식, half라고도 함
FLOAT_BFP16	특수 브레인 부동 소수점 형식

© 2000-2025, MetaQuotes Ltd.

이러한 각 형식에는 장점과 제한 사항이 있습니다. FLOAT16은 더 높은 정확도를 제공하지만 더 많은 저장 공간과 계산 리소스가 필요합니다. 반면 BFLOAT16은 데이터 처리에 더 높은 성능과 효율성을 제공하지만 정확도가 떨어질 수 있습니다.

ENUM_FLOAT8_FORMAT

ENUM_FLOAT8_FORMAT 열거형은 네 가지 FP8 유형 형식을 설명합니다.

FP8(8비트 부동 소수점)은 부동 소수점 자리를 나타내는 데 사용되는 데이터 유형 중 하나입니다. FP8에서 각 숫자는 8개의 데이터 비트로 표시되며 일반적으로 부호, 지수 및 가수의 세 가지 구성 요소로 나뉩니다. 이 형식은 정확성과 저장 효율성 사이의 균형을 제공하므로 메모리 용량과 계산 효율성이 필요한 애플리케이션에 적합합니다.

ID	설명
FLOAT_FP8_E4M3FN	8비트 부동 소수점 자리 숫자, 지수용 4비트, 가수용 3비트. 일반적으로 계수로 사용됩니다.
FLOAT_FP8_E4M3FNUZ	8비트 부동 소수점 자리 숫자, 지수용 4비트, 가수용 3비트. NaN을 지원하고 음의 0과 Inf를 지원하지 않습니다. 일반적으로 계수로 사용됩니다.
FLOAT_FP8_E5M2FN	8비트 부동 소수점 숫자, 지수용 5비트, 가수용 2비트. NaN과 Inf 지원. 일반적으로 기울기에 사용
FLOAT_FP8_E5M2FNUZ	8비트 부동 소수점 숫자, 지수용 5비트, 가수용 2비트. NaN을 지원하고 음의 0과 Inf를 지원하지 않습니다. 기울기에도 사용.

FP8의 주요 장점 중 하나는 대규모 데이터 세트 처리의 효율성입니다. FP8은 간결한 숫자 표현을 사용하여 메모리의 양을 줄이고 계산을 가속화합니다. 이는 대규모 데이터 세트를 자주 처리하는 기계 학습 및 인공 지능 애플리케이션에서 특히 중요합니다.

© 2000-2025, MetaQuotes Ltd.

스탠다드 라이브러리

이 챕터의 그룹에는 MQL5 표준 라이브러리에 대한 기술 세부 정보와 모든 주요 구성 요소에 대한 설명이 포함되어 있습니다.

MQL5 스탠다드 라이브러리는 MQL5로 작성되었으며 최종 사용자를 위한 프로그램(표시자, 스크립트, 전문가)을 쉽게 작성할 수 있도록 고안되었습니다. 라이브러리를 사용하면 대부분의 내부 MQL5 기능에 편리하게 액세스할 수 있습니다.

MQL5 스탠다드 라이브러리는 'Include' 폴더에 있는 터미널의 작업 디렉토리에 있습니다.

섹션	위치
수학	Include\Math\
OpenCL	Include\OpenCL\
기본 클래스 CObject	Include\
데이터 모음	Include\Arrays\
일반 데이터 모음	Include\Generic\
파일	Include\Files\
문자열	Include\Strings\
그래픽 개체	Include\Objects\
사용자 지정 그래픽	Include\Canvas\
3D 그래픽	Include\Canvas\
가격 차트	Include\Charts\
과학 차트	Include\Graphics\
지표	Include\Indicators\
트레이드 클래스	Include\Trade\
전략 모듈	Include\Expert\
패널 및 대화 상자	Include\Controls\

© 2000-2025, MetaQuotes Ltd.

수학

수학의 다양한 영역에서 계산을 수행하기 위해 여러 라이브러리가 제공됩니다:

- 통계 – 확률 이론의 다양한 분포를 다루는 함수
- 퍼지 로직 – 맘다니(Mamdani)와 스게노(Sugeno) 퍼지 추론 시스템을 구현하는 라이브러리
- ALGLIB – 데이터 분석 (클러스터링, 의사결정 트리, 선형 회귀 분석, 뉴럴 네트워크), 미분방정식 해결, 푸리에 변환, 수치적분, 최적화 문제, 통계 분석 등.

© 2000-2025, MetaQuotes Ltd.

统计

Statistical Library는 기초통계분포를 편리하게 작업할 수 있는 방법을 제공합니다.

라이브러리는 각 분포에 대해 5가지 기능을 제공합니다:

1. 확률 밀도 계산 – MathProbabilityDensityX() 형식의 함수
2. 확률 계산 – MathCumulativeDistributionX() 형식의 함수
3. 분포 분위수 계산 – MathQuantileX() 형식의 함수
4. 지정된 분포의 난수 생성 – MathRandomX() 형식의 함수
5. 분포의 이론적 모멘트 계산 – MathMomentsX() 형식의 함수

개별 무작위 변수의 값을 계산하는 것 외에도 라이브러리는 함수에 대한 오버로드도 제공하며, 이는 배열에 대해 동일한 계산을 수행합니다.

- 통계적 특성
- 정규 분포
- 로그 정규 분포
- 베타 분포
- 비중심 베타 분포
- 감마 분포
- 카이 제곱 분포
- 비중심 카이 제곱 분포
- 지수 분포
- F-분포
- 비중심 F-분포
- t-분포
- 비중심 t-분포
- 로지스틱 분포
- 코시 분포
- 균등 분포
- Weibull 분포
- 이항 분포
- 음의 이항 분포
- 기하 분포
- 초기하 분포
- 푸와송 분포
- 하위 함수

예:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                    NormalDistributionExample.mq5 |
//|                        Copyright 2016, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- 정규 분포를 계산하기 위한 함수를 포함
#include <Math\Stat\Normal.mqh>
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함                                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 정규 분포의 매개변수를 설정
   double mu=5.0;
   double sigma=1.0;
   PrintFormat("Normal distribution with parameters mu=%G and sigma=%G, calculation ex
//--- 간격 설정
   double x1=mu-sigma;
   double x2=mu+sigma;
//--- 확률 계산을 위한 변수
   double cdf1,cdf2,probability;
//--- 오류 코드를 위한 변수
   int error_code1,error_code2;
//--- 분포 함수의 값을 계산
   cdf1=MathCumulativeDistributionNormal(x1,mu,sigma,error_code1);
   cdf2=MathCumulativeDistributionNormal(x2,mu,sigma,error_code2);
//--- 오류 코드 확인
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- 범위 내의 무작위 변수의 확률 계산
      probability=cdf2-cdf1;
      //--- 결과 출력
      PrintFormat("1. Calculate probability of a random variable within the range of %
      PrintFormat("  Answer: Probability = %5.8f",probability);
     }

//--- 95% 신뢰 수준에 해당하는 랜덤 변수 x의 값 범위 찾기
   probability=0.95; //  신뢰 확률을 설정
//--- 확률을 구간 경계로 설정
   double p1=(1.0-probability)*0.5;
   double p2=probability+(1.0-probability)*0.5;
//--- 구간 경계를 계산
   x1=MathQuantileNormal(p1,mu,sigma,error_code1);
   x2=MathQuantileNormal(p2,mu,sigma,error_code2);
//--- 오류 코드 확인
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- 결과 출력
      PrintFormat("2. For confidence interval = %.2f, find the range of random variabl
      PrintFormat("  Answer: range is  %5.8f <= x <=%5.8f",x1,x2);
     }

   PrintFormat("3. Compute the first 4 calculated and theoretical moments of the distr
//--- 난수 배열을 생성하여 처음 네 모멘트를 계산하고 이론적 값과 비교
   int data_count=1000000;  // 값의 수 설정 및 배열 준비
   double data[];
   ArrayResize(data,data_count);
//--- 무작위 값을 생성하여 배열에 저장
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<data_count; i++)
     {
      data[i]=MathRandomNormal(mu,sigma,error_code1);
     }
//--- 초기값의 인덱스 및 산출을 위한 데이터의 양을 설정
   int start=0;
   int count=data_count;
//--- 생성된 값의 처음 네 모멘트를 계산
   double mean=MathMean(data,start,count);
   double variance=MathVariance(data,start,count);
   double skewness=MathSkewness(data,start,count);
   double kurtosis=MathKurtosis(data,start,count);
//--- 이론적 모멘트에 대한 변수
   double normal_mean=0;
   double normal_variance=0;
   double normal_skewness=0;
   double normal_kurtosis=0;
//--- 계산된 모멘트의 값을 표시
   PrintFormat("            Mean           Variance           Skewness           Kurtos
   PrintFormat("Calculated  %.10f    %.10f      %.10f      %.10f",mean,variance,skewnes
//--- 모멘트의 이론적 값을 계산하여 구한 값과 비교
   if(MathMomentsNormal(mu,sigma,normal_mean,normal_variance,normal_skewness,normal_ku
     {
      PrintFormat("Theoretical %.10f    %.10f      %.10f       %.10f",normal_mean,norma
      PrintFormat("Difference  %.10f    %.10f      %.10f       %.10f",mean-normal_mean,
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

통계적 특성

이 함수 그룹은 배열 요소의 통계적 특성을 계산합니다:

- 평균,
- 분산,
- 왜도,
- 첨도,
- 중위,
- 평균 제곱근 및
- 표준 편차.

함수	설명
MathMean	배열 요소의 평균(첫 번째 모멘트)을 계산하기
MathVariance	배열 요소의 평균(두 번째 모멘트)을 계산하기
MathSkewness	배열 요소의 왜도(세 번째 모멘트)를 계산하기
MathKurtosis	배열 요소의 첨도(네 번째 모멘트)를 계산하기
MathMoments	배열 요소의 처음 네 모멘트(평균, 분산, 왜도, 첨도)를 계산하기
MathMedian	배열 요소의 중위값 계산하기
MathStandardDeviation	배열 요소의 표준 편차 계산하기
MathAverageDeviation	배열 요소의 평균 절대 편차를 계산하기

© 2000-2025, MetaQuotes Ltd.

MathMean

배열 요소의 평균(첫 번째 모멘트)을 계산합니다. R의 mean()의 아날로그.

```
double  MathMean(
   const double&  array[]                // 배열(데이터 포함)
   );
```

매개변수

array

[in] 평균을 계산하기 위한 데이터가 있는 배열.

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

배열 요소의 평균. 오류가 발생한 경우 NaN(숫자 아님)을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathVariance

배열 요소의 분산(두 번째 모멘트)을 계산합니다. R의 var()의 아날로그.

```
double  MathVariance(
   const double&  array[]                // 배열(데이터 포함)
   );
```

매개변수

array

[in] 계산할 데이터가 있는 배열.

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

배열 요소의 분산. 오류가 발생한 경우 NaN(숫자 아님)을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathSkewness

배열 요소의 왜도(세 번째 모멘트)를 계산합니다. R의 skewness()의 아날로그(e1071 라이브러리).

```
double  MathSkewness(
   const double&  array[]                // 배열(데이터 포함)
   );
```

매개변수

array

[in] 계산할 데이터가 있는 배열.

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

배열 요소의 왜도. 오류가 발생한 경우 NaN(숫자 아님)을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathKurtosis

Calculates the kurtosis (fourth moment) of array elements. Analog of the kurtosis() in R (e1071 library).

```
double  MathKurtosis(
   const double&  array[]                // 배열(데이터 포함)
   );
```

매개변수

array

[in] 계산할 데이터가 있는 배열.

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

배열 요소의 첨도. 오류가 발생한 경우 NaN(숫자 아님)을 반환합니다.

Disclaimer

첨도의 계산은 정규 분포 주변의 초과 첨도를 사용하여 수행되며(초과 첨도=첨도-3), 즉 정규 분포의 초과 첨도는 0입니다.

기대값 주변의 분포 정점이 뾰족하면 양수이고, 정점이 평평하면 음수입니다.

© 2000-2025, MetaQuotes Ltd.

MathMoments

배열 요소의 처음 네 모멘트(평균, 분산, 왜도, 첨도)를 계산합니다

```
double  MathMoments(
   const double&  array[],              // 배열(데이터 포함)
   double&        mean,                 // 평균(첫 번째 모멘트)
   double&        variance,             // 분산(두 번째 모멘트)
   double&        skewness,             // 왜도(세 번째 모멘트)
   double&        kurtosis,             // 첨도(네 번째 모멘트)
   const int      start=0,              // 초기 인덱스
   const int      count=WHOLE_ARRAY     // 요소의 수
   );
```

매개변수

array

[in] 계산할 데이터가 있는 배열.

mean

[out] 평균에 대한 변수(첫 번째 모멘트)

variance

[out] 분산에 대한 변수(두 번째 모멘트)

skewness

[out] 왜도에 대한 변수(세 번째 모멘트)

kurtosis

[out] 첨도에 대한 변수(네 번째 모멘트)

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

모멘트가 성공적으로 계산되면 true를, 그렇지 않으면 false를 반환합니다.

Disclaimer

첨도의 계산은 정규 분포 주변의 초과 첨도를 사용하여 수행되며(초과 첨도=첨도-3), 즉 정규 분포의 초과 첨도는 0입니다.

기대값 주변의 분포 정점이 뾰족하면 양수이고, 정점이 평평하면 음수입니다.

© 2000-2025, MetaQuotes Ltd.

MathMedian

배열 요소의 중위값을 계산합니다. R의 median()의 아날로그.

```
double  MathMedian(
   const double&  array[]                // 배열(데이터 포함)
   );
```

매개변수

array

[in] 계산할 데이터가 있는 배열.

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

배열 요소의 중위 값. 오류가 발생한 경우 NaN(숫자 아님)을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathStandardDeviation

배열 요소의 표준 편차를 계산합니다. R의 sd()의 아날로그.

```
double  MathStandardDeviation(
   const double&  array[]                // 배열(데이터 포함)
   );
```

매개변수

array

[in] 계산할 데이터가 있는 배열.

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

배열 요소의 표준 편차. 오류가 발생한 경우 NaN(숫자 아님)을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathAverageDeviation

배열 요소의 평균 절대 편차를 계산합니다. R의 aad()의 아날로그.

```
double  MathAverageDeviation(
   const double&  array[]                // 배열(데이터 포함)
   );
```

매개변수

array

[in] 계산할 데이터가 있는 배열.

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

배열 요소의 평균 절대 편차. 오류가 발생한 경우 NaN(숫자 아님)을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

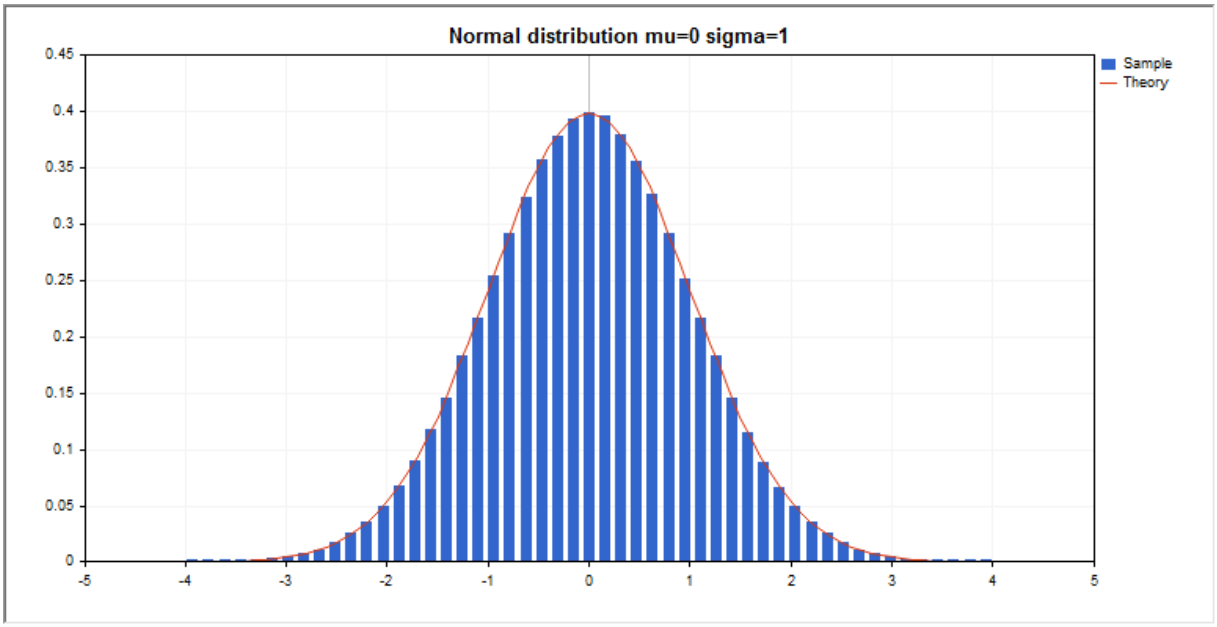
정규 분포

이 섹션에는 정규 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 정규 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Normal}(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

위치:

- x — 무작위 변수의 값
- μ — 기대값
- σ — 평균 제곱근 편차



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityNormal	정규 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionNormal	정규 확률 분포 함수의 값을 계산
MathQuantileNormal	지정된 확률에 대한 역 정규 분포 함수의 값을 계산
MathRandomNormal	정규 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 싱성
MathMomentsNormal	정규 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Normal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double mean_value=0;  // 기대값(평균)
input double std_dev=1;     // 평균 제곱근 편차(표준 편차)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // 표본의 값 수
   int ncells=51;        // 히스토그램의 구간 수
   double x[];           // 히스토그램 구간의 중심
   double y[];           // 구간 내에 포함되는 표본의 값 수
   double data[];        // 무작위 값의 표본
   double max,min;       // 표본의 최대값 및 최소값
//--- 정규 분포에서 표본 구함
   MathRandomNormal(mean_value,std_dev,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNormal(x2,mean_value,std_dev,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Normal distribution mu=%G sigma=%G",mean_value
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
//--- 모든 곡선 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
```

© 2000-2025, MetaQuotes Ltd.

```
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNormal

무작위 변수 x에 뮤 및 시그마 매개변수를 사용하여 정규 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNormal(
   const double  x,              // 무작위 변수 값
   const double  mu,             // 분포의 평균 매개변수(기대값)
   const double  sigma,          // 분포의 시그마 매개변수(평균 제곱근 편차)
   const bool    log_mode,       // 값의 로그 계산
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 뮤 및 시그마 매개변수를 사용하여 정규 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNormal(
   const double  x,              // 무작위 변수 값
   const double  mu,             // 분포의 평균 매개변수(기대값)
   const double  sigma,          // 분포의 시그마 매개변수(평균 제곱근 편차)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 정규 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dnorm()의 아날로그.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],           // 무작위 변수 값을 갖는 배열
   const double   mu,            // 분포의 평균 매개변수(기대값)
   const double   sigma,         // 분포의 시그마 매개변수(평균 제곱근 편차)
   const bool     log_mode,      // 값의 로그 계산
   double&        result[]       // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 정규 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],           // 무작위 변수 값을 갖는 배열
   const double   mu,            // 분포의 평균 매개변수(기대값)
   const double   sigma,         // 분포의 시그마 매개변수(평균 제곱근 편차)
   double&        result[]       // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

mu

[in] 분포의 평균 매개변수(기대값).

sigma

[in] 분포의 시그마 매개변수(평균 제곱근 편차).

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 확률 밀도 함수의 값을 얻을 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNormal

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 정규 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // 무작위 변수 값
   const double  mu,             // 기대값
   const double  sigma,          // 평균 제곱근 편차
   const bool    tail,           // 테일 계산용 플래그
   const bool    log_mode,       // 값의 로그 계산
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 정규 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // 무작위 변수 값
   const double  mu,             // 기대값
   const double  sigma,          // 평균 제곱근 편차
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 정규 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dnorm()의 아날로그.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 기대값
   const double  sigma,          // 평균 제곱근 편차
   const bool    tail,           // 테일 계산용 플래그
   const bool    log_mode,       // 값의 로그 계산
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 정규 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 기대값
   const double  sigma,          // 평균 제곱근 편차
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

mu

[in] 분포의 평균 매개변수(기대값).

sigma

[in] 분포의 시그마 매개변수(평균 제곱근 편차).

tail

[in] 계산 플래그. tail=true이면 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 확률 함수의 값을 얻을 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNormal

지정된 *확률*에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 정규 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNormal(
   const double  probability,    // 무작위 변수의 확률 값
   const double  mu,             // 기대값
   const double  sigma,          // 평균 제곱근 편차
   const bool    tail,           // 테일 계산용 플래그
   const bool    log_mode,       // 값의 로그 계산
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 정규 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNormal(
   const double  probability,    // 무작위 변수의 확률 값
   const double  mu,             // 기대값
   const double  sigma,          // 평균 제곱근 편차
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 뮤 및 시그마 매개변수를 사용하여 역 정규 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qnorm()의 아날로그.

```
bool  MathQuantileNormal(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  mu,             // 기대값
   const double  sigma,          // 평균 제곱근 편차
   const bool    tail,           // 테일 계산용 플래그
   const bool    log_mode,       // 값의 로그 계산
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 뮤 및 시그마 매개변수를 사용하여 역 정규 분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileNormal(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  mu,             // 기대값
   const double  sigma,          // 평균 제곱근 편차
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

mu

[in] 분포의 평균 매개변수(기대값).

sigma

[in] 분포의 시그마 매개변수(평균 제곱근 편차).

tail

[in] 계산 플래그. false이면 1.0 확률에 대한 계산이 수행됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수를 얻을 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomNormal

뮤 및 시그마 매개변수를 사용하여 정규 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomNormal(
   const double  mu,              // 기대값
   const double  sigma,           // 평균 제곱근 편차
   int&          error_code       // 오류 코드를 저장할 변수
   );
```

뮤 및 시그마 매개변수를 사용하여 정규 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rnorm()의 아날로그.

```
bool  MathRandomNormal(
   const double  mu,              // 기대값
   const double  sigma,           // 평균 제곱근 편차
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수를 얻는 배열
   );
```

매개변수

mu

[in] 분포의 평균 매개변수(기대값).

sigma

[in] 분포의 시그마 매개변수(평균 제곱근 편차).

data_count

[in] 얻을 의사 난수 변수의 수.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNormal

정규 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsNormal(
   const double  mu,              // 기대값
   const double  sigma,           // 평균 제곱근 편차
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드를 저장할 변수
   );
```

매개변수

mu

[in] 분포의 평균 매개변수(기대값).

sigma

[in] 분포의 시그마 매개변수(평균 제곱근 편차).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트가 성공적으로 계산되면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

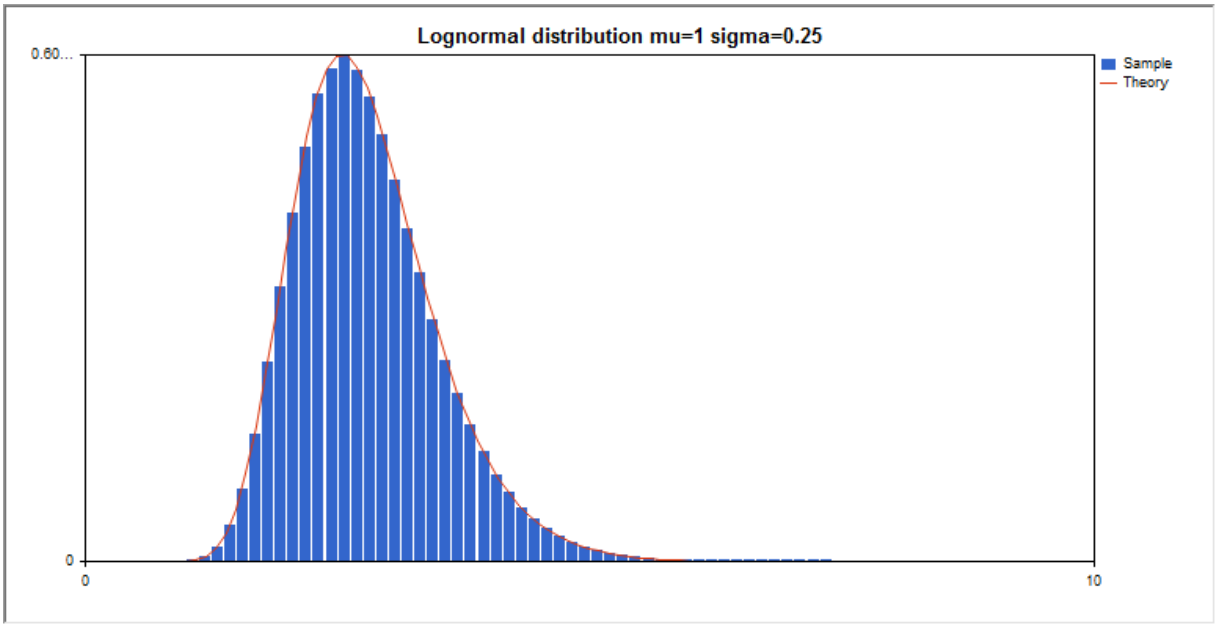
로그 정규 분포

이 섹션에는 로그 정규 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 로그 정규 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 로그 정규 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Lognormal}(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

위치:

- x — 무작위 변수의 값
- μ — 기대값의 로그
- σ — 평균 제곱근 편차의 로그



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityLognormal	로그 정규 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionLognormal	로그 정규 확률 분포 함수의 값을 계산
MathQuantileLognormal	지정된 확률에 대한 역 로그 정규 분포 함수의 값을 계산
MathRandomLognormal	로그 정규 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsLognormal	로그 정규 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Lognormal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double mean_value=1.0;  // 기대값의 로그(로그 평균)
input double std_dev=0.25;    // 평균 제곱근 편차의 로그(로그 표준 편차)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // 표본의 값 수
   int ncells=51;          // 히스토그램의 구간 수
   double x[];             // 히스토그램 구간의 중심
   double y[];             // 구간 내에 포함되는 표본의 값 수
   double data[];          // 무작위 값의 표본
   double max,min;         // 표본의 최대값 및 최소값
//--- 로그 정규 분포에서 표본 구함
   MathRandomLognormal(mean_value,std_dev,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLognormal(x2,mean_value,std_dev,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Lognormal distribution mu=%G sigma=%G",mean_va
   graphic.BackgroundMainSize(16);
//--- Y 축의 자동 스케일링 비활성화
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityLognormal

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 로그 정규 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // 무작위 변수 값
   const double  mu,             // 기대값의 로그(로그 평균)
   const double  sigma,          // 평균 제곱근 편차의 로그(로그 표준 편차)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 로그 정규 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // 무작위 변수 값
   const double  mu,             // 기대값의 로그(로그 평균)
   const double  sigma,          // 평균 제곱근 편차의 로그(로그 표준 편차)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 로그 정규 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다. R의 dlnorm()의 아날로그.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // 무작위 변수 값을 갖는 배열
   const double   mu,            // 기대값의 로그(로그 평균)
   const double   sigma,         // 평균 제곱근 편차의 로그(로그 표준 편차)
   const bool     log_mode,      // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로
   double&        result[]       // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 로그 정규 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // 무작위 변수 값을 갖는 배열
   const double   mu,            // 기대값의 로그(로그 평균)
   const double   sigma,         // 평균 제곱근 편차의 로그(로그 표준 편차)
   double&        result[]       // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

mu

[in] 기대 값의 로그 (log_mean).

sigma

[in] 평균 제곱근 편차(로그 표준 편차).

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수의 값을 얻을 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionLognormal

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 확률의 로그 정규 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // 무작위 변수 값
   const double  mu,             // 기대값의 로그(로그 평균)
   const double  sigma,          // 평균 제곱근 편차의 로그(로그 표준 편차)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값을 로그를 계산하는 log_mode=true면 확률의 자연 로그가
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 확률의 로그 정규 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // 무작위 변수 값
   const double  mu,             // 기대값의 로그(로그 평균)
   const double  sigma,          // 평균 제곱근 편차의 로그(로그 표준 편차)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 확률의 로그 정규 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 plnorm()의 아날로그.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // 무작위 변수 값을 갖는 배열
   const double   mu,            // 기대값의 로그(로그 평균)
   const double   sigma,         // 평균 제곱근 편차의 로그(로그 표준 편차)
   const bool     tail,          // 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확
   const bool     log_mode,      // 값의 로그를 계산하는 플래그 log_mode=true면 확률이 자
   double&        result[]       // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 확률의 로그 정규 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // 무작위 변수 값을 갖는 배열
   const double   mu,            // 기대값의 로그(로그 평균)
   const double   sigma,         // 평균 제곱근 편차의 로그(로그 표준 편차)
   double&        result[]       // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

mu

[in] 기대 값의 로그 (log_mean).

sigma

[in] 평균 제곱근 편차(로그 표준 편차).

tail

[in] 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값을 얻을 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileLognormal

지정된 확률에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 로그 정규 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileLognormal(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  mu,             // 기대값의 로그(로그 평균)
   const double  sigma,          // 평균 제곱근 편차의 로그(로그 표준 편차)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 확률에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 로그 정규 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileLognormal(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  mu,             // 기대값의 로그(로그 평균)
   const double  sigma,          // 평균 제곱근 편차의 로그(로그 표준 편차)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 probability[] 배열에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 로그 정규 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qlnorm()의 아날로그.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double   mu,             // 기대값의 로그(로그 평균)
   const double   sigma,          // 평균 제곱근 편차의 로그(로그 표준 편차)
   const bool     tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩
   const bool     log_mode,       // 계산 플래그 log_mode=true이면 Exp(확률)에 대해 계산이
   double&        result[]        // 백분위수 값을 갖는 배열
   );
```

확률 값의 지정된 probability[] 배열에 대해 뮤 및 시그마 매개변수를 사용하여 역 로그 정규 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double   mu,             // 기대값의 로그(로그 평균)
   const double   sigma,          // 평균 제곱근 편차의 로그(로그 표준 편차)
   double&        result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수 발생 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

mu

[in] 기대 값의 로그 (log_mean).

sigma

[in] 평균 제곱근 편차(로그 표준 편차).

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomLognormal

뮤 시그마 매개변수를 사용하여 로그 정규 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomLognormal(
   const double  mu,              // 기대값의 로그(로그 평균)
   const double  sigma,           // 평균 제곱근 편차의 로그(로그 표준 편차)
   int&          error_code       // 오류 코드를 저장할 변수
   );
```

뮤 시그마 매개변수를 사용하여 로그 정규 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rlnorm()의 아날로그.

```
double  MathRandomLognormal(
   const double  mu,              // 기대값의 로그(로그 평균)
   const double  sigma,           // 평균 제곱근 편차의 로그(로그 표준 편차)
   const int     data_count,      // 필요 데이터 양
   double&       result[]         // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

mu

[in] 기대 값의 로그 (log_mean).

sigma

[in] 평균 제곱근 편차(로그 표준 편차).

data_count

[in] 필요한 데이터의 양.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 의사 난수 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsLognormal

로그 정규 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다. 모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

```
double  MathMomentsLognormal(
   const double  mu,              // 기대값의 로그(로그 평균)
   const double  sigma,           // 평균 제곱근 편차의 로그(로그 표준 편차)
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드를 저장할 변수
   );
```

매개변수

mu

[in] 기대 값의 로그 (log_mean).

sigma

[in] 평균 제곱근 편차(로그 표준 편차).

mean

[in] 평균에 대한 변수.

variance

[out] 분산에 대한 변수.

skewness

[out] 왜도에 대한 변수.

kurtosis

[out] 첨도에 대한 변수.

error code

[out] 오류 코드를 저장할 변수.

값 반환

모멘트가 성공적으로 계산되면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

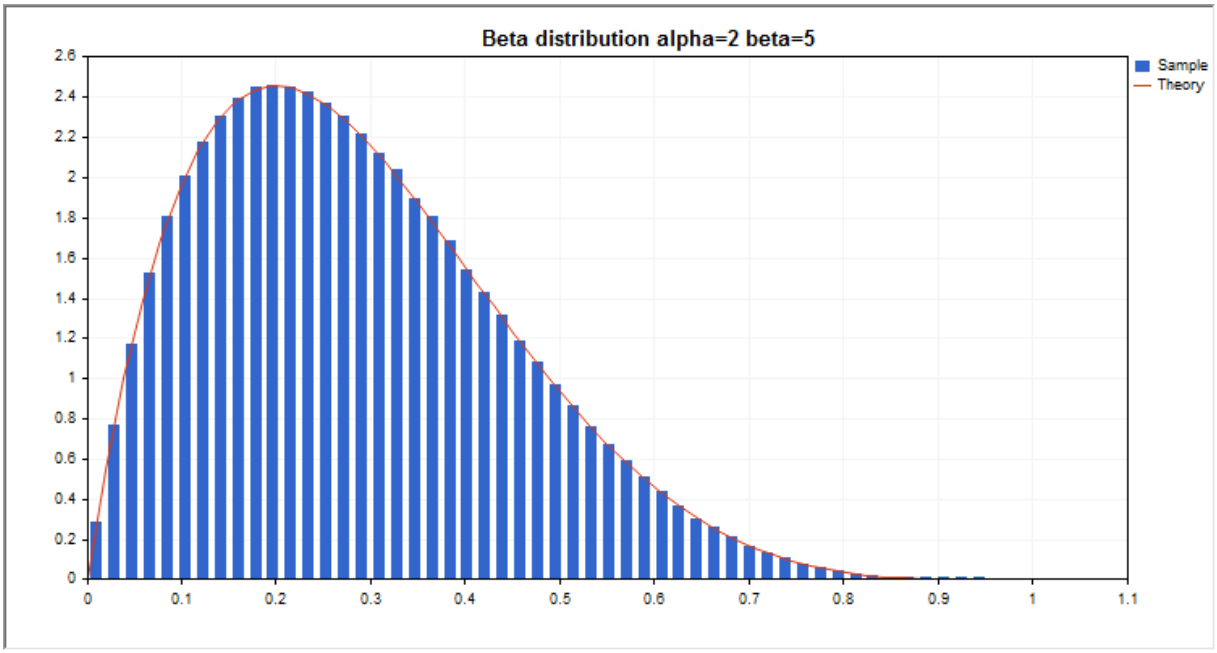
베타 분포

이 섹션에서는 베타 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 해당 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 베타 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Beta}(x \mid a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

위치:

- x — 무작위 변수의 값
- a — 베타 분포의 첫 번째 매개변수
- b — 베타 분포의 두 번째 매개변수



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityBeta	베타 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionBeta	베타 확률 분포 함수의 값을 계산
MathQuantileBeta	지정된 확률에 대한 역 베타 분포 함수의 값을 계산
MathRandomBeta	베타 분포 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsBeta	베타 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Beta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double alpha=2;   // 베타 분포의 첫 번째 매개변수(형상1)
input double beta=5;    // 베타 분포의 두 번째 매개변수(형상2)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=51;       // 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 베타 분포에서 표본 구함
   MathRandomBeta(alpha,beta,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityBeta(x2,alpha,beta,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Beta distribution alpha=%G beta=%G",alpha,beta
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
```

© 2000-2025, MetaQuotes Ltd.

```
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityBeta

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 베타 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityBeta(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 베타 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityBeta(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 베타 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dbeta()의 아날로그.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 베타 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

a

[in] 베타 분포의 첫 번째 매개변수(형상1).

b

[in] 베타 분포의 두 번째 매개변수(형상2)

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionBeta

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 베타 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확률
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 베타 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 베타 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 pbeta()의 아날로그.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const bool    tail,           // 계산된 플래그가 true면 무작위 변수가 x를 초과하지 않을
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 지
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

<t0>무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 베타 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

a

[in] 베타 분포의 첫 번째 매개변수(형상1).

b

[in] 베타 분포의 두 번째 매개변수(형상2)

tail

[in] 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileBeta

지정된 *확률*에 대해 함수는 a 및 b 매개변수를 사용하여 역 베타 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileBeta(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 a 및 b 매개변수를 사용하여 역 베타 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileBeta(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 함수는 a 및 b 매개변수를 사용하여 역 베타 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qbeta()의 아날로그.

```
double  MathQuantileBeta(
   const double&  probability[], // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 함수는 a 및 b 매개변수를 사용하여 역 베타 분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileBeta(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

a

[in] 베타 분포의 첫 번째 매개변수(형상1).

b

[in] 베타 분포의 두 번째 매개변수(형상2).

tail

[in] lower_tail=false면 계산 플래그가 1.0 확률에 대해 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomBeta

a 및 b 매개변수를 사용하여 베타 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomBeta(
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

a 및 b 매개변수를 사용하여 베타 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성하기. 오류인 경우 false를 반환합니다. R의 rbeta()의 아날로그.

```
bool  MathRandomBeta(
   const double  a,               // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,               // 베타 분포의 두 번째 매개변수(형상2)
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수를 얻는 배열
   );
```

매개변수

a

[in] 베타 분포의 첫 번째 매개변수(형상1)

b

[in] 베타 분포의 두 번째 매개변수(형상2).

data_count

[in] 얻을 의사 난수 변수의 수.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsBeta

베타 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsBeta(
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

a

[in] 베타 분포의 첫 번째 매개변수(형상1).

b

[in] 베타 분포의 두 번째 매개변수(형상2).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트가 성공적으로 계산되면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

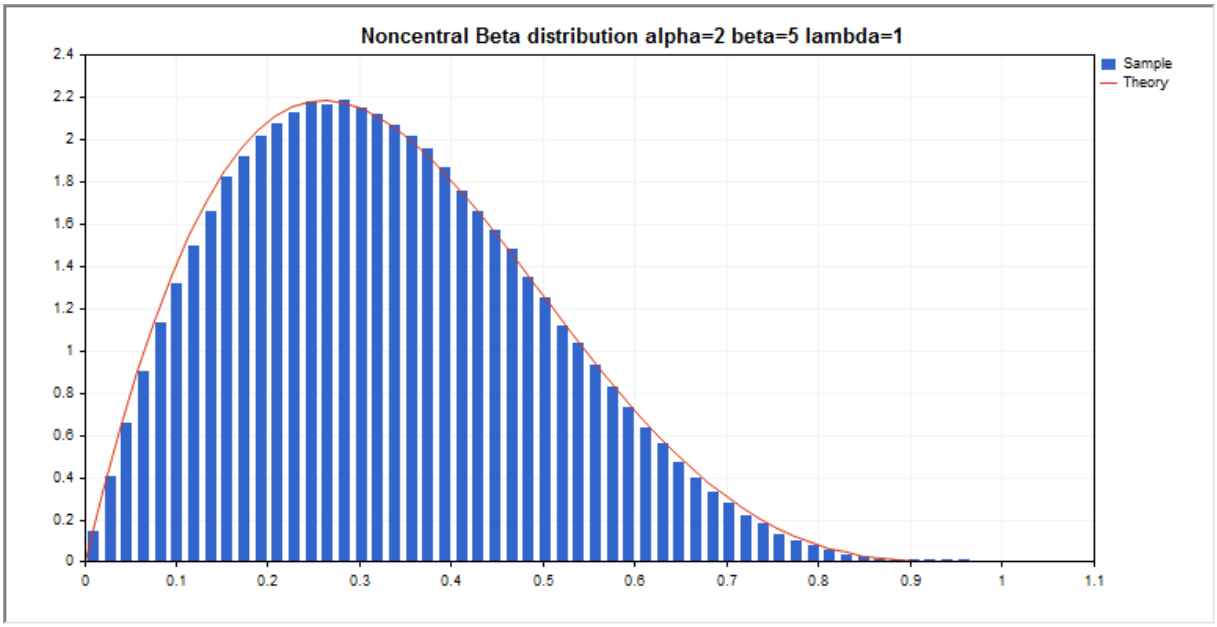
비중심 베타 분포

이 섹션에서는 비중심 베타 배포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 해당 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 비중심 베타 분포는 다음 공식으로 정의됩니다:

$$f_{NoncentralBeta}(x|a,b,\lambda)=\sum_{r=0}^{\infty}e^{-\frac{\lambda}{2}}\frac{\left(\frac{\lambda}{2}\right)^{r}}{r!}\frac{x^{a+r-1}(1-x)^{b-1}}{B(a+r,b)}$$

위치:

- x — 무작위 변수의 값
- a — 베타 분포의 첫 번째 매개변수
- b — 베타 분포의 두 번째 매개변수
- λ — noncentrality parameter



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityNoncentralBeta	비중심 베타 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionNoncentralBeta	비중심 베타 확률 분포 함수의 값을 계산
MathQuantileNoncentralBeta	지정된 확률에 대한 역 비중심 베타 분포 함수의 값을 계산
MathRandomNoncentralBeta	비중심 베타 분포 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성

© 2000-2025, MetaQuotes Ltd.

함수	설명
MathMomentsNoncentralBeta	비중심 베타 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralBeta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double a_par=2;     // 베타 분포의 첫 번째 매개변수(형상1)
input double b_par=5;     // 베타 분포의 두 번째 매개변수(형상2)
input double l_par=1;     // 비중심 매개변수(람다)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // 표본의 값 수
   int ncells=53;        // the number of intervals in the histogram
   double x[];           // 히스토그램 구간의 중심
   double y[];           // 구간 내에 포함되는 표본의 값 수
   double data[];        // 무작위 값의 표본
   double max,min;       // 표본의 최대값 및 최소값
//--- 비중심 베타 분포에서 표본 구함
   MathRandomNoncentralBeta(a_par,b_par,l_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralBeta(x2,a_par,b_par,l_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

© 2000-2025, MetaQuotes Ltd.

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral Beta distribution alpha=%G beta=%G
                          a_par,b_par,l_par));
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
    return (true);
   }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
   {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
    double range=MathAbs(maxv-minv);
    int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
    maxv=NormalizeDouble(maxv,degree);
    minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
    stepv=NormalizeDouble(MathPow(10,-degree),degree);
    if((maxv-minv)/stepv<10)
       stepv/=10.;
   }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralBeta

무작위 변수 x에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dbeta()의 아날로그.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

a

[in] 베타 분포의 첫 번째 매개변수(형상1).

b

[in] 베타 분포의 두 번째 매개변수(형상2)

lambda

[in] 비중심 매개변수

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralBeta

무작위 변수 x에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확률
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 pbeta()의 아날로그.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   const bool    tail,           // 계산 플래그가 true인 경우 무작위 변수가 x를 초과하지 않을
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

© 2000-2025, MetaQuotes Ltd.

매개변수

x
[in] 무작위 변수의 값.

x[]
[in] 무작위 변수 값이 있는 배열.

a
[in] 베타 분포의 첫 번째 매개변수(형상1).

b
[in] 베타 분포의 두 번째 매개변수(형상2)

lambda
[in] 비중심 매개변수

tail
[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode
[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code
[out] 오류 코드를 저장할 변수.

result[]
[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralBeta

무작위 변수 *확률*의 발생 확률에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 역 확률 분포 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 *확률*의 발생 확률에 대한 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 역 확률 분포 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 probability[] 배열에 대해 함수는 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 역 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qbeta()의 아날로그.

```
double  MathQuantileNoncentralBeta(
   const double&  probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,               // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,               // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,          // 비중심 매개변수
   const bool    tail,            // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,        // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]         // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 probability[] 배열에 대해 함수는 a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 역 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathQuantileNoncentralBeta(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

© 2000-2025, MetaQuotes Ltd.

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

[in] 무작위 변수의 확률 값을 갖는 배열.

a

[in] 베타 분포의 첫 번째 매개변수(형상1).

b

[in] 베타 분포의 두 번째 매개변수(형상2).

lambda

[in] 비중심 매개변수.

tail

[in] 계산 플래그 false안 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] 계산 플래그 log_mode=true면 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralBeta

비중심 베타 분포의 법칙 a, b 및 람다 매개변수에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomNoncentralBeta(
   const double  a,             // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,             // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,        // 비중심 매개변수
   int&          error_code     // 오류 코드를 저장할 변수
   );
```

비중심 베타 분포의 법칙에 따라 분포된 의사 난수 변수 a, b 및 람다 매개변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rbeta()의 아날로그.

```
bool  MathRandomNoncentralBeta(
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수를 얻는 배열
   );
```

매개변수

a

[in] 베타 분포의 첫 번째 매개변수(형상1)

b

[in] 베타 분포의 두 번째 매개변수(형상2).

lambda

[in] 비중심 매개변수

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralBeta

a, b 및 람다 매개변수를 사용하여 비중심 베타 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsNoncentralBeta(
   const double  a,              // 베타 분포의 첫 번째 매개변수(형상1)
   const double  b,              // 베타 분포의 두 번째 매개변수(형상2)
   const double  lambda,         // 비중심 매개변수
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

a

[in] 베타 분포의 첫 번째 매개변수(형상1).

b

[in] 베타 분포의 두 번째 매개변수(형상2).

lambda

[in] 비중심 매개변수

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

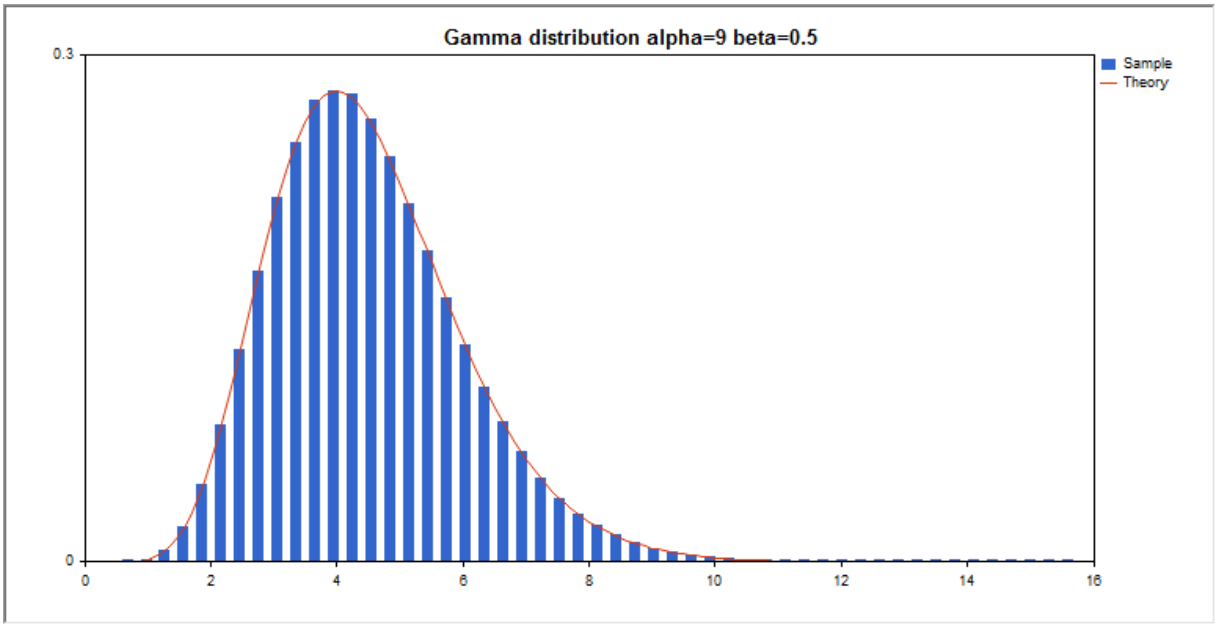
감마 분포

이 섹션에서는 감마 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 해당 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 감마 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Gamma}(x|a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}}$$

위치:

- x — 무작위 변수의 값
- a — 분포의 첫 번째 매개변수
- b — 분포의 두 번째 매개변수



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityGamma	감마 분포의 확률 밀도 함수를 계산합니다
MathCumulativeDistributionGamma	감마 확률 분포 함수의 값을 계산합니다
MathQuantileGamma	지정된 확률에 대한 역 감마 분포 함수의 값을 계산합니다
MathRandomGamma	감마 분포 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성합니다
MathMomentsGamma	감마 분포의 처음 네 모멘트의 이론적 수치 값을 계산합니다

예:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Gamma.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double alpha=9;   // 분포의 첫 번째 매개변수(형상)
input double beta=0.5;  // 분포의 두 번째 매개변수(스케일)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=51;       // 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 감마 분포에서 표본 구함
   MathRandomGamma(alpha,beta,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityGamma(x2,alpha,beta,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Gamma distribution alpha=%G beta=%G",alpha,bet
   graphic.BackgroundMainSize(16);
//--- Y 축의 자동 스케일링 비활성화
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,1));
   graphic.YAxis().Min(0);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityGamma

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 감마 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityGamma(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 감마 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityGamma(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 감마 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dgamma()의 아날로그.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 감마 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

a

[in] 분포의 첫 번째 매개변수(형상).

b

[in] 분포의 두 번째 매개변수(스케일).

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionGamma

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 감마 분포의 확률 분포 함수를 계한삽니다 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 감마 분포의 확률 분포 함수를 계한삽니다 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 감마 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 pgamma()의 아날로그.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확률
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 감마 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

a

[in] 분포의 첫 번째 매개변수(형상).

b

[in] 분포의 두 번째 매개변수(스케일)

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileGamma

지정된 *확률*에 대해 함수 a 및 b 매개변수를 사용하여 역 감마 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileGamma(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수 a 및 b 매개변수를 사용하여 역 감마 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileGamma(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 a 및 b 매개변수를 사용하여 역 감마 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qgamma()의 아날로그.

```
double  MathQuantileGamma(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 a 및 b 매개변수를 사용하여 역 감마 분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileGamma(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

a

[in] 분포의 첫 번째 매개변수(형상).

b

[in] 분포의 두 번째 매개변수(스케일).

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomGamma

a 및 b 매개변수를 사용하여 감마 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomGamma(
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

a 및 b 매개변수를 사용하여 감마 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rgamma()의 아날로그.

```
bool  MathRandomGamma(
   const double  a,               // 분포의 첫 번째 매개변수(형상)
   const double  b,               // 분포의 두 번째 매개변수(스케일)
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

a

[in] 분포의 첫 번째 매개변수(형상).

b

[in] 분포의 두 번째 매개변수(스케일).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsGamma

a 및 b 매개변수를 사용하여 감마 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsGamma(
   const double  a,              // 분포의 첫 번째 매개변수(형상)
   const double  b,              // 분포의 두 번째 매개변수(스케일)
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

a

[in] 분포의 첫 번째 매개변수(형상).

b

[in] 분포의 두 번째 매개변수(스케일).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

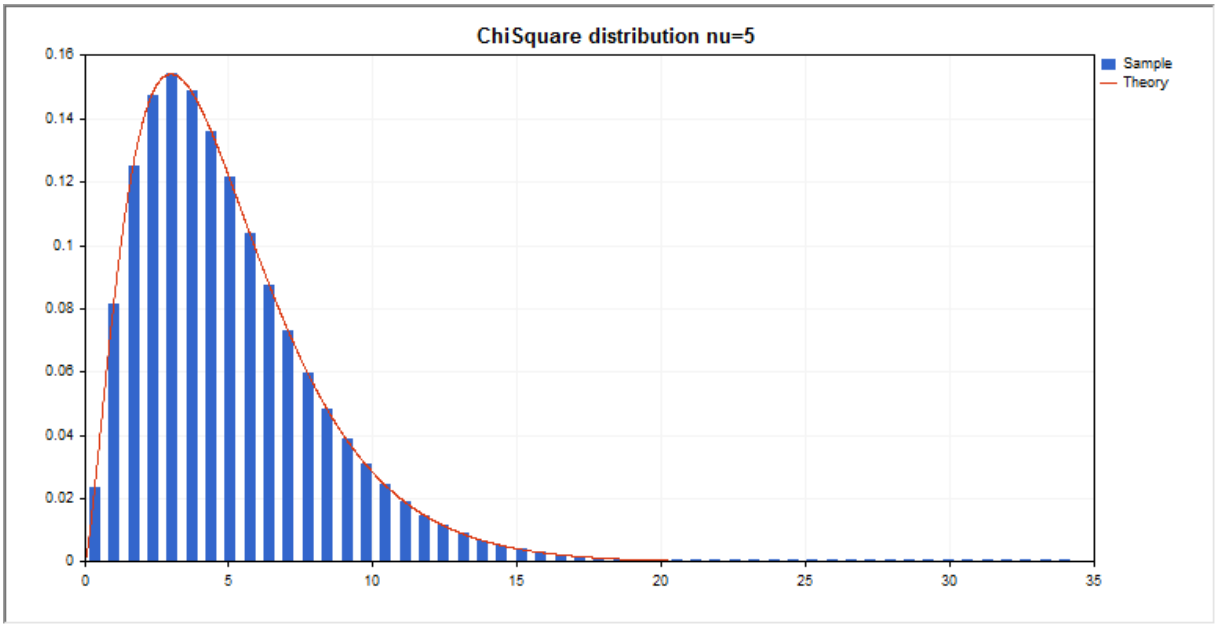
카이 제곱 분포

이 섹션에서는 카이 제곱 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 해당 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 카이 제곱 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Chi-Square}(x|\nu) = \frac{x^{\frac{(\nu-2)}{2}} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)}$$

위치:

- x — 무작위 변수의 값
- ν — 자유도



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityChiSquare	카이 제곱 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionChiSquare	카이 제곱 확률 분포 함수의 값을 계산
MathQuantileChiSquare	지정된 확률에 대한 역 카이 제곱 분포 함수의 값을 계산
MathRandomChiSquare	카이 제곱 분포 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsChiSquare	카이 제곱 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

© 2000-2025, MetaQuotes Ltd.

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\ChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double nu_par=5;    // 자유도
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=51;       // 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 카이 제곱 분포에서 표본 구함
   MathRandomChiSquare(nu_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityChiSquare(x2,nu_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
```

© 2000-2025, MetaQuotes Ltd.

```
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("ChiSquare distribution nu=%G ",nu_par));
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
```

© 2000-2025, MetaQuotes Ltd.

```
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityChiSquare

무작위 변수 x에 대한 뉴 매개변수를 사용하여 카이 제곱 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴 매개변수를 사용하여 카이 제곱 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 카이 제곱 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dchisq()의 아날로그.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 카이 제곱 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

nu

[in] 분포 매개변수(자유도)

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] 値の ログを 計算する フラグ. log_mode=true面 確率 密度の 自然 ログが 返されます.

error_code

[out] エラー コードを 格納する 変数.

result[]

[out] 確率 密度 関数 値の 配列.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionChiSquare

무작위 변수 x에 대한 뉴 매개변수를 사용하여 카이 제곱 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴 매개변수를 사용하여 카이 제곱 분포 함수를 계산합니다. In case of error it returns NaN

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 카이 제곱 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 pchisq()의 아날로그.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확률
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 카이 제곱 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

nu

[in] 분포 매개변수(자유도).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileChiSquare

지정된 *확률*에 대해 함수는 역 카이 제곱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileChiSquare(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu,             // 분포 매개변수(자유도)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 역 카이 제곱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileChiSquare(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu,             // 분포 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 함수는 역 카이 제곱 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qchisq()의 아날로그.

```
double  MathQuantileChiSquare(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 함수는 역 카이 제곱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileChiSquare(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

[in] 무작위 변수의 확률 값을 갖는 배열.

nu

[in] 분포 매개변수(자유도).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomChiSquare

뉴 매개변수를 사용하여 카이 제곱 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomChiSquare(
   const double  nu,            // 분포 매개변수(자유도)
   int&          error_code     // 오류 코드를 저장할 변수
   );
```

뉴 매개변수를 사용하여 카이 제곱 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rchisq()의 아날로그.

```
bool  MathRandomChiSquare(
   const double  nu,             // 분포 매개변수(자유도)
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

nu

[in] 분포 매개변수(자유도).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsChiSquare

뉴 매개변수를 사용하여 카이 제곱 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsChiSquare(
   const double  nu,              // 분포 매개변수(자유도)
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드에 대한 변수
   );
```

매개변수

nu

[in] 분포 매개변수(자유도).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

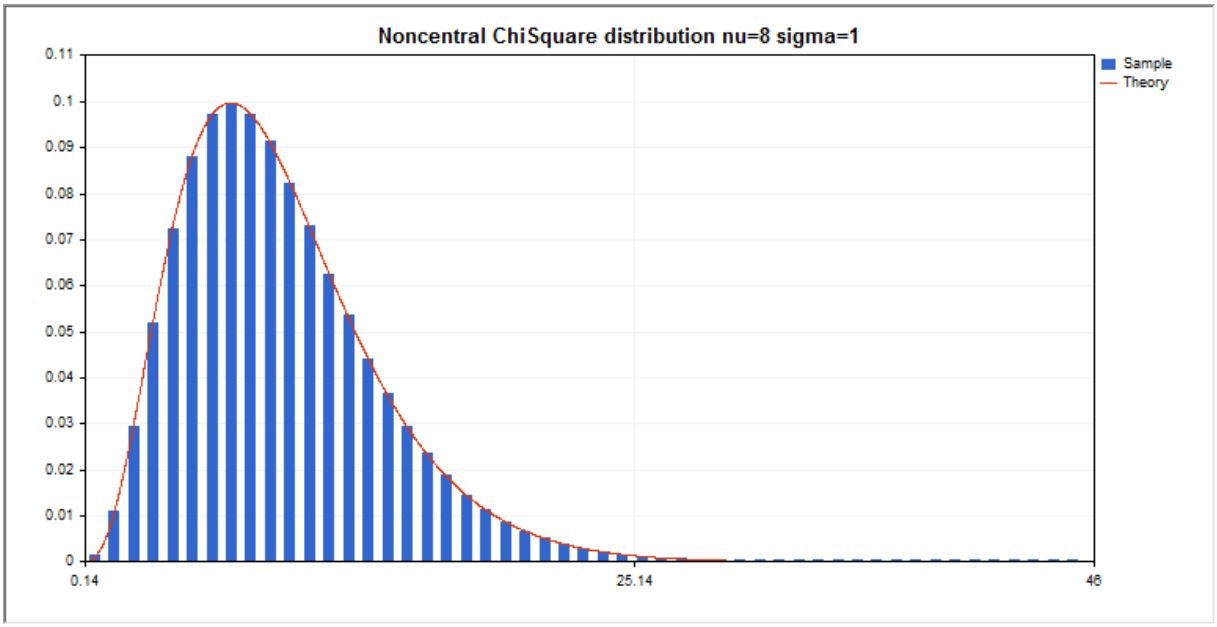
비중심 카이 제곱 분포

이 섹션에는 비중심 카이 제곱 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 해당 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 비중심 카이 제곱 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{NoncentralChiSquare}(x|\nu,\sigma)=\frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{1}{2}\right)}x^{\frac{\nu}{2}-1}e^{-\frac{(x+\sigma)}{2}}\sum_{r=0}^{\infty}\frac{(\lambda x)^r}{(2r)!}\frac{\Gamma\left(\frac{1}{2}+r\right)}{\Gamma\left(\frac{\nu}{2}+r\right)}$$

위치:

- x — 무작위 변수의 값
- ν — 자유도
- σ — 비중심 매개변수



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityNoncentralChiSquare	비중심 카이 제곱 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionNoncentralChiSquare	비중심 카이 제곱 확률 분포 함수의 값을 계산
MathQuantileNoncentralChiSquare	지정된 확률에 대한 역 비중심 카이 제곱 분포 함수의 값을 계산
MathRandomNoncentralChiSquare	비중심 카이 제곱 분포 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성

© 2000-2025, MetaQuotes Ltd.

함수	설명
MathMomentsNoncentralChiSquare	비중심 카이 제곱 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double nu_par=8;     // 자유도
input double si_par=1;     // 비중심 매개변수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // 표본의 값 수
   int ncells=51;          // 히스토그램의 구간 수
   double x[];             // 히스토그램 구간의 중심
   double y[];             // 구간 내에 포함되는 표본의 값 수
   double data[];          // 무작위 값의 표본
   double max,min;         // 표본의 최대값 및 최소값
//--- 비중심 카이 제곱 분포에서 표본 구함
   MathRandomNoncentralChiSquare(nu_par,si_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralChiSquare(x2,nu_par,si_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
```

© 2000-2025, MetaQuotes Ltd.

```
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral ChiSquare distribution nu=%G sigma=
   graphic.BackgroundMainSize(16);
//--- X 축의 자동 스케일링 비활성화
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(NormalizeDouble(max,0));
   graphic.XAxis().Min(min);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

© 2000-2025, MetaQuotes Ltd.

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralChiSquare

무작위 변수 x에 대한 뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dchisq()의 아날로그.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 비중심 카이 제곱 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

nu

[in] 분포 매개변수(자유도).

sigma

[in] 비중심 매개변수.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralChiSquare

무작위 변수 x에 대한 뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 lower_tail=true면 무작위 변수 x를 초과하지
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 pchisq()의 아날로그.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 lower_tail=true면 x를 무작위 변수 x를 초과
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

nu

[in] 분포 매개변수(자유도).

sigma

[in] 비중심 매개변수.

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralChiSquare

지정된 *확률*에 대해 함수는 뉴 및 시그마 매개변수를 사용하여 역 비중심 카이 제곱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 뉴 및 시그마 매개변수를 사용하여 역 비중심 카이 제곱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뉴 및 시그마 매개변수를 사용하여 역 비중심 카이 제곱 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qchisq()의 아날로그.

```
double  MathQuantileNoncentralChiSquare(
   const double&  probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu,              // 분포 매개변수(자유도)
   const double  sigma,           // 비중심 매개변수
   const bool    tail,            // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩
   const bool    log_mode,        // 계싼 플래그 log_mode=true면 Exp(확률)에 대해 계산이 =
   double&       result[]         // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 역 비중심 카이 제곱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileNoncentralChiSquare(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

nu

[in] 분포 매개변수(자유도).

sigma

[in] 비중심 매개변수.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralChiSquare

뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomNoncentralChiSquare(
   const double  nu,            // 분포 매개변수(자유도)
   const double  sigma,         // 비중심 매개변수
   int&          error_code     // 오류 코드를 저장할 변수
   );
```

뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rchisq()의 아날로그.

```
bool  MathRandomNoncentralChiSquare(
   const double  nu,              // 분포 매개변수(자유도)
   const double  sigma,           // 비중심 매개변수
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

nu

[in] 분포 매개변수(자유도).

sigma

[in] 비중심 매개변수.

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralChiSquare

뉴 및 시그마 매개변수를 사용하여 비중심 카이 제곱 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsNoncentralChiSquare(
   const double  nu,              // 분포 매개변수(자유도)
   const double  sigma,           // 비중심 매개변수
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드에 대한 변수
   );
```

매개변수

nu

[in] 분포 매개변수(자유도).

sigma

[in] 비중심 매개변수.

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

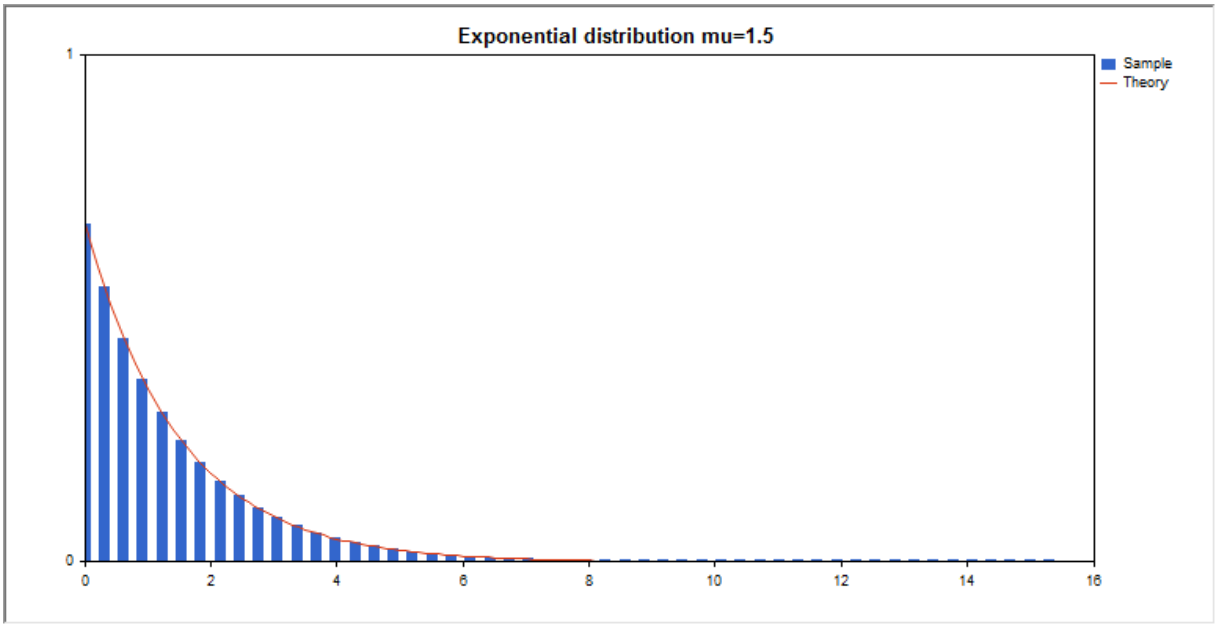
지수

이 섹션에서는 지수 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 지수 분포의 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 지수 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Exponential}(x|\mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

위치:

- x — 무작위 변수의 값
- μ — 기대값



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityExponential	지수 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionExponential	지수 확률 분포 함수의 값을 계산
MathQuantileExponential	지정된 확률에 대한 역 지수 분포 함수의 값을 계산
MathRandomExponential	지수 분포의 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsExponential	지수 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

```
#include <Graphics\Graphic.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Math\Stat\Exponential.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double mu_par=1.5;     // the number of degrees of freedom
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // 표본의 값 수
   int ncells=51;          // 히스토그램의 구간 수
   double x[];             // 히스토그램 구간의 중심
   double y[];             // 구간 내에 포함되는 표본의 값 수
   double data[];          // 무작위 값의 표본
   double max,min;         // 표본의 최대값 및 최소값
//--- 지수 분포에서 표본 구함
   MathRandomExponential(mu_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityExponential(x2,mu_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Exponential distribution mu=%G ",mu_par));
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
```

© 2000-2025, MetaQuotes Ltd.

```
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityExponential

무작위 변수 x에 대한 뮤 매개변수를 사용하여 지수 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityExponential(
   const double  x,              // 무작위 변수의 값
   const double  mu,             // 분포 매개변수(기대값)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뮤 매개변수를 사용하여 지수 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityExponential(
   const double  x,              // 무작위 변수의 값
   const double  mu,             // 분포 매개변수(기대값)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뮤 매개변수를 사용하여 지수 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dexp()의 아날로그.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 분포 매개변수(기대값)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뮤 매개변수를 사용하여 지수 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 분포 매개변수(기대값)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

mu

[in] 분포 매개변수(기대값)

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] 値の ログを 計算する フラグ. log_mode=true面 確率 密度の 自然 ログが 返されます.

error_code

[out] エラー コードを 保存する 変数.

result[]

[out] 確率 密度 関数 値の 配列.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionExponential

무작위 변수 x에 대한 뮤 매개변수를 사용하여 확률의 지수 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // 무작위 변수의 값
   const double  mu,             // 분포 매개변수(기대값)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뮤 매개변수를 사용하여 확률의 지수 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // 무작위 변수의 값
   const double  mu,             // 분포 매개변수(기대값)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뮤 매개변수를 사용하여 확률의 지수 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 pexp()의 아날로그.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 분포 매개변수(기대값)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확률
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뮤 매개변수를 사용하여 확률의 지수 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 분포 매개변수(기대값)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

mu

[in] 분포 매개변수(기대값).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileExponential

지정된 *확률*에 대해 함수는 뮤 매개변수를 사용하여 역 지수 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileExponential(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  mu,             // 분포 매개변수(기대값)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 뮤 매개변수를 사용하여 역 지수 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileExponential(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  mu,             // 분포 매개변수(기대값)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뮤 매개변수를 사용하여 역 지수 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qexp()의 아날로그.

```
double  MathQuantileExponential(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  mu,             // 분포 매개변수(기대값)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뮤 매개변수를 사용하여 역 지수 분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileExponential(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  mu,             // 분포 매개변수(기대값)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

[in] 무작위 변수의 확률 값을 갖는 배열.

mu

[in] 분포 매개변수(기대값).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomExponential

뮤 매개변수를 사용하여 지수 분포 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomExponential(
   const double  mu,             // 분포 매개변수(기대값)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

뮤 매개변수를 사용하여 지수 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rexp()의 아날로그.

```
bool  MathRandomExponential(
   const double  mu,              // 분포 매개변수(기대값)
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

mu

[in] 분포 매개변수(기대값).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsExponential

뮤 매개변수를 사용하여 지수 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsExponential(
   const double  mu,              // 분포 매개변수(기대값)
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드에 대한 변수
   );
```

매개변수

mu

[in] 분포의 매개변수(기대값).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

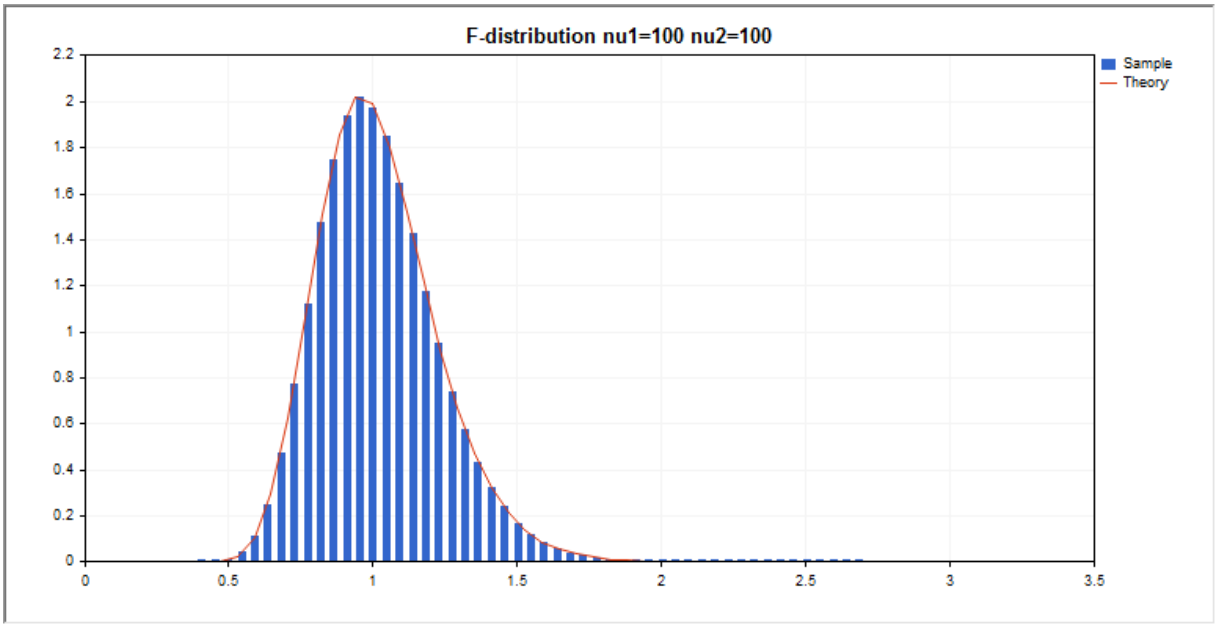
F-분포

이 섹션에서는 F-분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 피셔의 F-분포의 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. F-분포는 다음과 같은 공식으로 정의됩니다 :

$$f_F(x|\nu_1,\nu_2)=\frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}\right)}{\Gamma\left(\frac{\nu_1}{2}\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}}\frac{x^{\frac{\nu_1-2}{2}}}{\left(1+\left(\frac{\nu_1}{\nu_2}\right)x\right)^{\frac{\nu_1+\nu_2}{2}}}$$

위치:

- x — 무작위 변수의 값
- ν1 — 분포의 첫 번째 매개변수(자유도)
- ν2 — 분포의 두 번째 매개변수(자유도)



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityF	F-분포의 확률 밀도 함수를 계산
MathCumulativeDistributionF	F-분포 함수의 값을 계산
MathQuantileF	지정된 확률에 대한 역 F-분포 함수의 값을 계산
MathRandomF	피셔의 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열 생성
MathMomentsF	피셔의 F-분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\F.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double nu_1=100;    // 자유도의 첫 번째 수
input double nu_2=100;    // 자유도의 두 번째 수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // 표본의 값 수
   int ncells=51;          // 히스토그램의 구간 수
   double x[];             // 히스토그램 구간의 중심
   double y[];             // 구간 내에 포함되는 표본의 값 수
   double data[];          // 무작위 값의 표본
   double max,min;         // 표본의 최대값 및 최소값
//--- 피셔의 F-분포에서 표본 구함
   MathRandomF(nu_1,nu_2,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityF(x2,nu_1,nu_2,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("F-distribution nu1=%G nu2=%G",nu_1,nu_2));
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(4);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
```

© 2000-2025, MetaQuotes Ltd.

```
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityF

무작위 변수 x에 대한 뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityF(
   const double  x,              // 무작위 변수의 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityF(
   const double  x,              // 무작위 변수의 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 df()의 아날로그.

```
bool  MathProbabilityDensityF(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityF(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

nu1

[in] 분포의 첫 번째 매개변수(자유도).

nu2

[in] 분포의 두 번째 매개변수(자유도).

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionF

무작위 변수 x에 대한 뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 확률 분포 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionF(
   const double  x,              // 무작위 변수의 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 확률 분포 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionF(
   const double  x,              // 무작위 변수의 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 확률 분포 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 pf()의 아날로그.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확률
   const bool    log_mode,       // 값을 로그를 계산하는 log_mode=true면 확률의 자연 로그가
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 확률 분포 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

nu1

[in] 분포의 첫 번째 매개변수(자유도).

nu2

[in] 분포의 두 번째 매개변수(자유도).

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileF

지정된 *확률*에 대해 함수는 뉴1 및 뉴2 매개변수를 사용하여 역 F-분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileF(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 뉴1 및 뉴2 매개변수를 사용하여 역 F-분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileF(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *probability[]* 배열에 대해 함수는 뉴1 및 뉴2 매개변수를 사용하여 역 F-분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qf()의 아날로그.

```
double  MathQuantileF(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

지정된 *probability[]* 배열에 대해 함수는 뉴1 및 뉴2 매개변수를 사용하여 역 F-분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileF(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

nu1

[in] 분포의 첫 번째 매개변수(자유도).

nu2

[in] 분포의 두 번째 매개변수(자유도).

tail

[in] lower_tail=false면 계산 플래그가 1.0 확률에 대해 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomF

뉴1 및 뉴2 매개변수를 사용하여 피셔의 F-분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomF(
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

뉴1 및 뉴2 매개변수를 사용하여 피셔의 F-분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rf()의 아날로그.

```
bool  MathRandomF(
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

nu1

[in] 분포의 첫 번째 매개변수(자유도).

nu2

[in] 분포의 두 번째 매개변수(자유도).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsF

뉴1 및 뉴2 매개변수를 사용하여 피셔 F-분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsF(
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

nu1

[in] 분포의 첫 번째 매개변수(자유도).

nu2

[in] 분포의 두 번째 매개변수(자유도).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

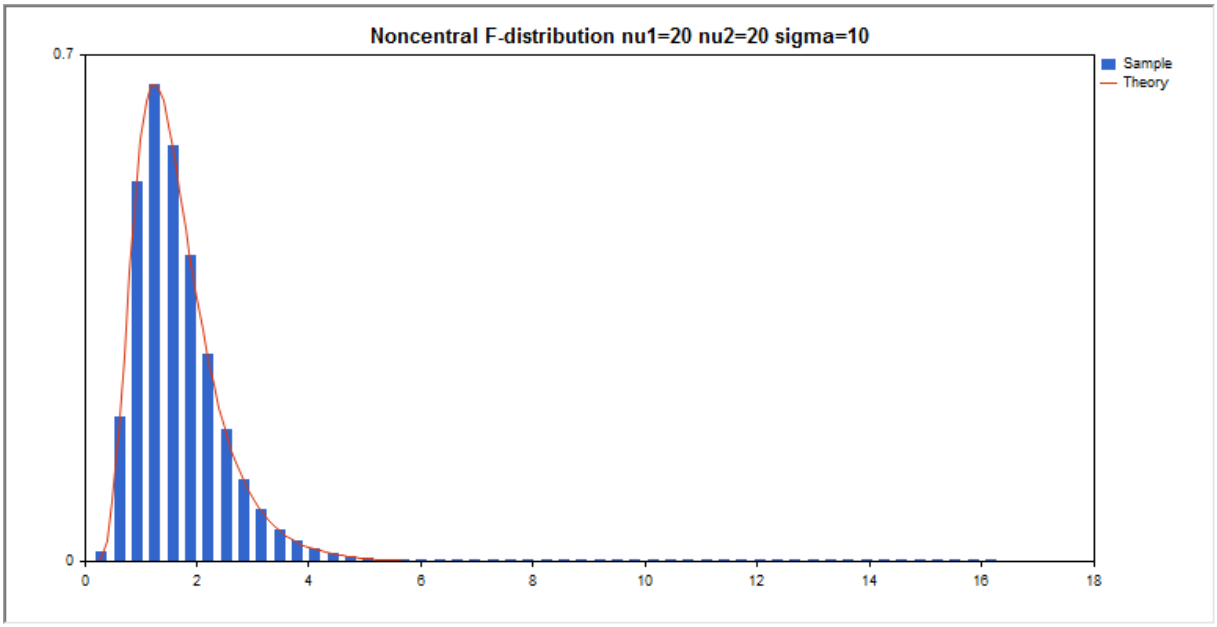
비중심 F-분포

이 섹션에 비중심 F-분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 비중심 피셔 F-분포의 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 비중심 F-분포는 다음과 같은 공식으로 정의됩니다:

$$f_{NoncentralF}(x|\nu_1,\nu_2,\sigma) = e^{-\frac{\sigma}{2}}\sum_{r=0}^{\infty}\frac{1}{r!}\left(\frac{\sigma}{2}\right)^r \frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}+r\right)}{\Gamma\left(\frac{\nu_2}{2}+r\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_2}{2}+r} \frac{x^{\frac{\nu_2}{2}-1+r}}{\left(1+\frac{\nu_1}{\nu_2}x\right)^{\frac{\nu_1+\nu_2}{2}+r}}$$

위치:

- x — 무작위 변수의 값
- ν1 — 분포의 첫 번째 매개변수(자유도)
- ν2 — 분포의 두 번째 매개변수(자유도)
- σ — 비중심 매개변수



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityNoncentralF	비중심 F-분포의 확률 밀도 함수를 계산
MathCumulativeDistributionNoncentralF	비중심 F-분포 함수의 값을 계산
MathQuantileNoncentralF	지정된 확률에 대한 역 비중심 F-분포 함수의 값을 계산
MathRandomNoncentralF	비중심 피셔 분포 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성

© 2000-2025, MetaQuotes Ltd.

함수	설명
MathMomentsNoncentralF	비중심 F-분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralF.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double nu_1=20;    // 자유도의 첫 번째 수
input double nu_2=20;    // 자유도의 두 번째 수
input double sig=10;     // 비중심 매개변수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=51;       // 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 비중심 피셔의 F-분포에서 표본 구함
   MathRandomNoncentralF(nu_1,nu_2,sig,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralF(x2,nu_1,nu_2,sig,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

© 2000-2025, MetaQuotes Ltd.

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral F-distribution nu1=%G nu2=%G sigma=
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralF

무작위 변수 x에 대한 뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 F-분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // 무작위 변수의 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 F-분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // 무작위 변수의 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 F-분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 df()의 아날로그.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 F-분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

nu1

[in] 분포의 첫 번째 매개변수(자유도).

nu2

[in] 분포의 두 번째 매개변수(자유도).

sigma

[in] 비중심 매개변수.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralF

무작위 변수 x에 대한 뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 F-분포의 확률 분포 함수 값을 계산합니다 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // 무작위 변수의 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 F-분포의 확률 분포 함수 값을 계산합니다 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // 무작위 변수의 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[]에 대한 뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 피셔 F-분포의 확률 분포 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 pf()의 아날로그.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴1, 뉴2 및 시그마 매개변수를 사용하여 중심적이지 않은 피셔 F-분포의 확률 분포 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

© 2000-2025, MetaQuotes Ltd.

매개변수

x
[in] 무작위 변수의 값.

x[]
[in] 무작위 변수 값이 있는 배열.

nu1
[in] 분포의 첫 번째 매개변수(자유도).

nu2
[in] 분포의 두 번째 매개변수(자유도).

sigma
[in] 비중심 매개변수.

tail
[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode
[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code
[out] 오류 코드를 저장할 변수.

result[]
[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileF

지정된 *확률*에 대해 함수는 뉴1, 뉴2 및 시그마 매개변수를 사용하여 역 비중심 피셔의 F-분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileF(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 뉴1, 뉴2 및 시그마 매개변수를 사용하여 역 비중심 피셔의 F-분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileF(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뉴1, 뉴2 및 시그마 매개변수를 사용하여 역 비중심 피셔의 F-분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qf()의 아날로그.

```
double  MathQuantileF(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뉴1, 뉴2 및 시그마 매개변수를 사용하여 역 비중심 피셔의 F-분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileF(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

© 2000-2025, MetaQuotes Ltd.

probability
[in] 무작위 변수의 확률 값.

probability[]
[in] 무작위 변수의 확률 값을 갖는 배열.

nu1
[in] 분포의 첫 번째 매개변수(자유도).

nu2
[in] 분포의 두 번째 매개변수(자유도).

sigma
[in] 비중심 매개변수.

tail
[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode
[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code
[out] 오류 코드를 가져올 변수.

result[]
[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralF

뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 피셔의 F-분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomNoncentralF(
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

뉴1, 뉴2 및 시그마 매개변수를 사용하여 비중심 피셔 F-분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rf()의 아날로그.

```
bool  MathRandomNoncentralF(
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

nu1
  [in] 분포의 첫 번째 매개변수(자유도).

nu2
  [in] 분포의 두 번째 매개변수(자유도).

sigma
  [in] 비중심 매개변수.

error_code
  [out] 오류 코드를 저장할 변수.

data_count
  [out] 필요한 데이터의 양.

result[]
  [out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralF

뉴1, 뉴2 및 시그마 매개변수를 사용하여 중심적이지 않은 F-분포의 처음 네 모멘트의 이론적 수치 값을 계산합니다.

```
double  MathMomentsNoncentralF(
   const double  nu1,            // 분포의 첫 번째 매개변수(자유도)
   const double  nu2,            // 분포의 두 번째 매개변수(자유도)
   const double  sigma,          // 비중심 매개변수
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

nu1
  [in] 분포의 첫 번째 매개변수(자유도).

nu2
  [in] 분포의 두 번째 매개변수(자유도).

sigma
  [in] 비중심 매개변수.

mean
  [out] 평균 값을 가져올 변수.

variance
  [out] 분산을 가져올 변수.

skewness
  [out] 왜도를 가져올 변수.

kurtosis
  [out] 첨도를 가져올 변수.

error_code
  [out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

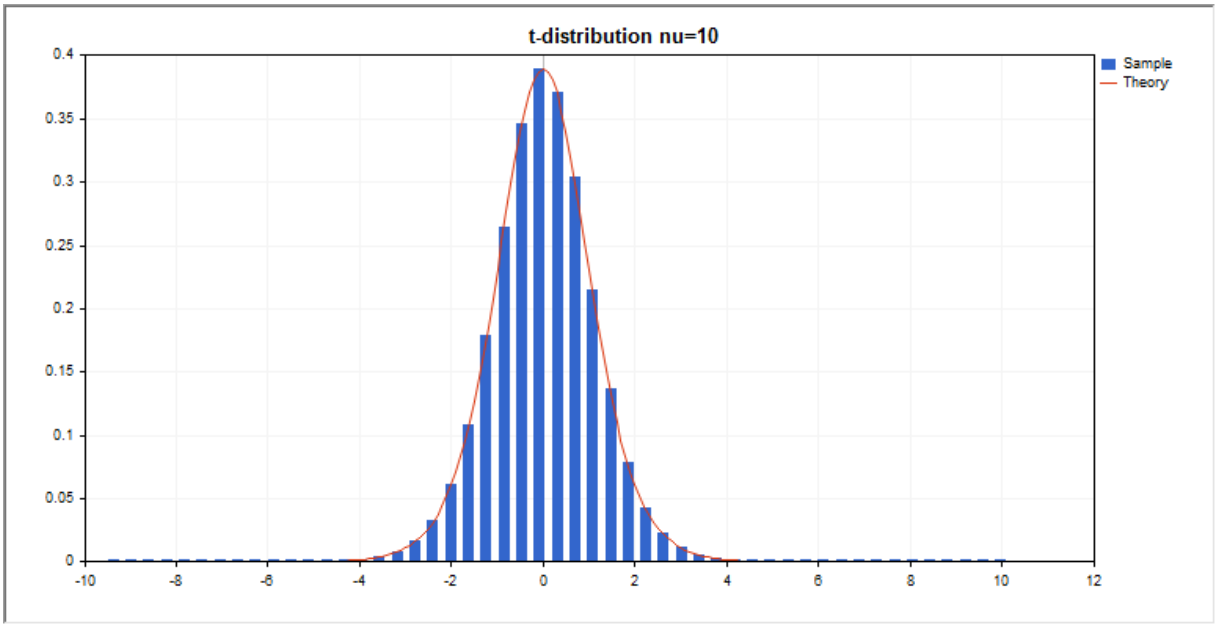
t-분포

이 섹션에는 스튜던트 t-분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 스튜던트 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. t-분포는 다음과 같은 공식으로 정의됩니다:

$$f_T(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \frac{1}{\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}}$$

위치:

- x — 무작위 변수의 값
- ν — 분포 매개변수(자유도)



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityT	t-분포의 확률 밀도 함수를 계산
MathCumulativeDistributionT	t-분포 함수의 값을 계산
MathQuantileT	지정된 확률에 대한 역 t-분포 함수의 값을 계산
MathRandomT	스튜던트 T-분포의 법칙에 따라 분포된 의사 난수 변수/의사 난수 배열을 생성
MathMomentsT	스튜던트 t-분포의 처음 네 모멘트의 이론적 수치 값을 계산

예:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\T.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double nu_par=10;    // 자유도
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=51;       // 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 스튜던트 t-분포에서 표본 구함
   MathRandomT(nu_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityT(x2,nu_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("t-distribution nu=%G",nu_par));
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityT

무작위 변수 x에 대한 뉴 매개변수를 사용하여 스튜던트의 t-분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityT(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴 매개변수를 사용하여 스튜던트의 t-분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityT(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 스튜던트 t-분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dt()의 아날로그.

```
bool  MathProbabilityDensityT(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 스튜던트 t-분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityT(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

nu

[in] 분포 매개변수(자유도).

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] 値の ログを 計算する フラグ. log_mode=true面 確率 密度の 自然 ログが 返されます.

error_code

[out] エラー コードを 保存する 変数.

result[]

[out] 確率 密度 関数 値の 配列.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionT

무작위 변수 x에 대한 뉴 매개변수를 사용하여 학생의 t-분포 함수 값을 계산합니다 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionT(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴 매개변수를 사용하여 학생의 t-분포 함수 값을 계산합니다 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionT(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 스튜던트의 t-분포 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 pt()의 아날로그.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴 매개변수를 사용하여 스튜던트의 t-분포 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

nu

[in] 분포 매개변수(자유도).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileT

지정된 *확률*에 대해 함수는 뉴 매개변수를 사용하여 역 스튜던트 t-분포 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileT(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu,             // 분포 매개변수(자유도)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 뉴 매개변수를 사용하여 역 스튜던트 t-분포 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileT(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu,             // 분포 매개변수(자유도)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뉴 매개변수를 사용하여 역 스튜던트 t-분포 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qt()의 아날로그.

```
double  MathQuantileT(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뉴 매개변수를 사용하여 역 스튜던트 t-분포 함수 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileT(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

[in] 무작위 변수의 확률 값을 갖는 배열.

nu

[in] 분포 매개변수(자유도).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomT

뉴 매개변수를 사용하여 스튜던트 t-분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomT(
   const double  nu,              // 분포 매개변수(자유도)
   int&          error_code       // 오류 코드를 저장할 변수
   );
```

뉴 매개변수를 사용하여 스튜던트 t-분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rt()의 아날로그.

```
bool  MathRandomT(
   const double  nu,              // 분포 매개변수(자유도)
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

nu

[in] 분포 매개변수(자유도).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsT

뉴 매개변수를 사용하여 스튜던트 t-분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsT(
   const double  nu,              // 분포 매개변수(자유도)
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드에 대한 변수
   );
```

매개변수

nu

[in] 분포 매개변수(자유도).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

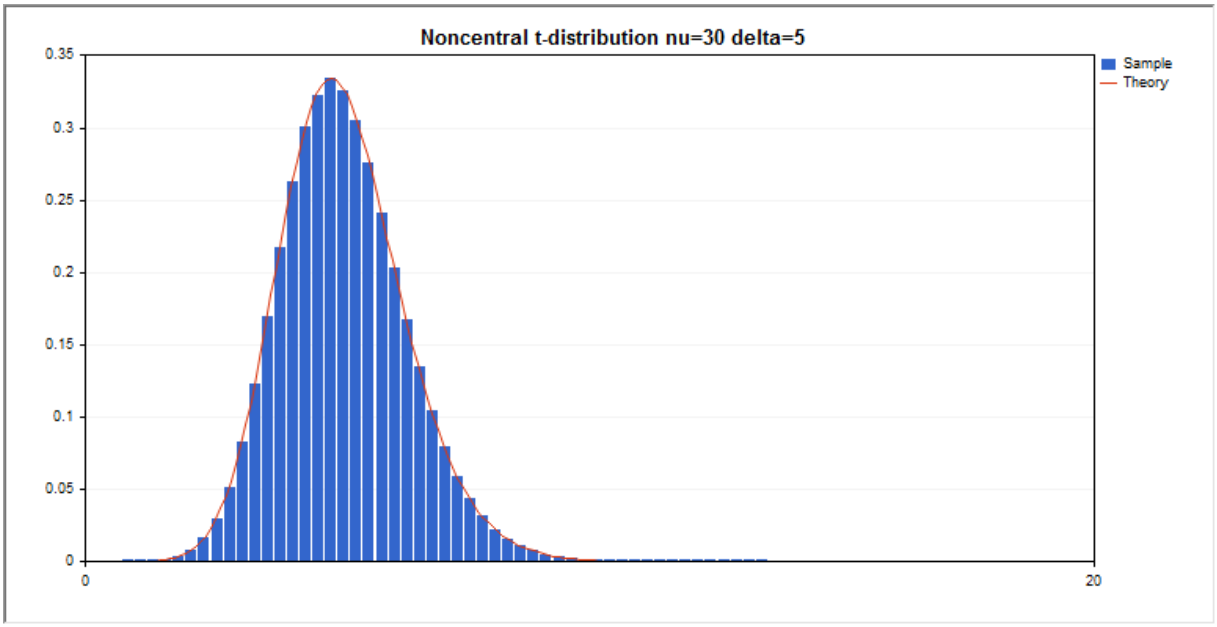
t-분포

이 섹션에는 비중심 스튜던트 t-분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 비중심 t-분포의 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 비중심 t-분포는 다음과 같은 공식으로 정의됩니다:

$$f_{NoncentralT}(x|\nu,\delta)=\frac{\nu^{\frac{\nu}{2}}e^{-\frac{\sigma^2}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\nu+x^2\right)^{\frac{\nu+1}{2}}}\sum_{r=0}^{\infty}\frac{(x\delta)^r}{r!}\left(\frac{2}{\nu+x^2}\right)^{\frac{r}{2}}\Gamma\left(\frac{\nu+r+1}{2}\right)$$

위치:

- x —무작위 변수의 값
- ν — 분포 매개변수(자유도)
- σ —비중심 매개변수



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityNoncentralT	비중심 t-분포의 확률 밀도 함수를 계산
MathCumulativeDistributionNoncentralT	비중심 t-분포 함수의 값을 계산
MathQuantileNoncentralT	지정된 확률에 대한 역 비중심 t-분포 함수의 값을 계산
MathRandomNoncentralT	비중심 스튜던트 t-분포 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsNoncentralT	비중심 스튜던트 t-분포의 처음 네 모멘트의 이론적 수치 값을 계산

© 2000-2025, MetaQuotes Ltd.

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralT.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double nu_par=30;        // 자유도
input double delta_par=5;     // 비중심 매개변수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // 표본의 값 수
   int ncells=51;        // 히스토그램의 구간 수
   double x[];           // 히스토그램 구간의 중심
   double y[];           // 구간 내에 포함되는 표본의 값 수
   double data[];        // 무작위 값의 표본
   double max,min;       // 표본의 최대값 및 최소값
//--- 비중심 스튜던트 t-분포에서 표본 구함
   MathRandomNoncentralT(nu_par,delta_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralT(x2,nu_par,delta_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Noncentral t-distribution nu=%G delta=%G",nu_p
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNoncentralT

무작위 변수 x에 대한 뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드 변수
   );
```

무작위 변수 x에 대한 뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   int&          error_code      // 오류 코드 변수
   );
```

무작위 변수 x[] 배열에 다한 뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dt()의 아날로그.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-매개변수의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

nu

[in] 분포 매개변수(자유도).

delta

[in] 비중심 매개변수.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNoncentralT

무작위 변수 x에 대한 뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // 무작위 변수의 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 pt()의 아날로그.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

nu

[in] 분포 매개변수(자유도).

delta

[in] 비중심 매개변수.

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNoncentralT

지정된 *확률*에 대해 함수는 뉴 및 델타 매개변수를 사용하여 역 비중심 학생 t-분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNoncentralT(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 lower_tail=false이면 1.0 확률에 대해 계산
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 뉴 및 델타 매개변수를 사용하여 역 비중심 학생 t-분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNoncentralT(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뉴 및 델타 매개변수를 사용하여 역 비중심 스튜던트 t-분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qt()의 아날로그.

```
double  MathQuantileNoncentralT(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   const bool    tail,           // 계산 플래그 lower_tail=false이면 1.0 확률에 대해 계산
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 뉴 및 델타 매개변수를 사용하여 역 비중심 스튜던트 t-분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileNoncentralT(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  nu,             // 분포 매개변수(자유도)
   const double  delta,          // 비중심 매개변수
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

nu

[in] 분포 매개변수(자유도).

delta

[in] 비중심 매개변수.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomNoncentralT

뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomNoncentralT(
   const double  nu,              // 분포 매개변수(자유도)
   const double  delta,           // 비중심 매개변수
   int&          error_code       // 오류 코드를 저장할 변수
   );
```

뉴 및 델타 매개변수를 사용하여 비중심 스튜던트 t-분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rt()의 아날로그.

```
bool  MathRandomNoncentralT(
   const double  nu,              // 분포 매개변수(자유도)
   const double  delta,           // 비중심 매개변수
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

nu

[in] 분포 매개변수(자유도).

delta

[in] 비중심 매개변수.

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNoncentralT

뉴 및 델타 매개변수를 사용하여 중심적이지 않은 스튜던트 t-분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsNoncentralT(
   const double  nu,              // 분포 매개변수(자유도)
   const double  delta,           // 비중심 매개변수
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드에 대한 변수
   );
```

매개변수

nu

[in] 분포 매개변수(자유도).

delta

[in] 비중심 매개변수.

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

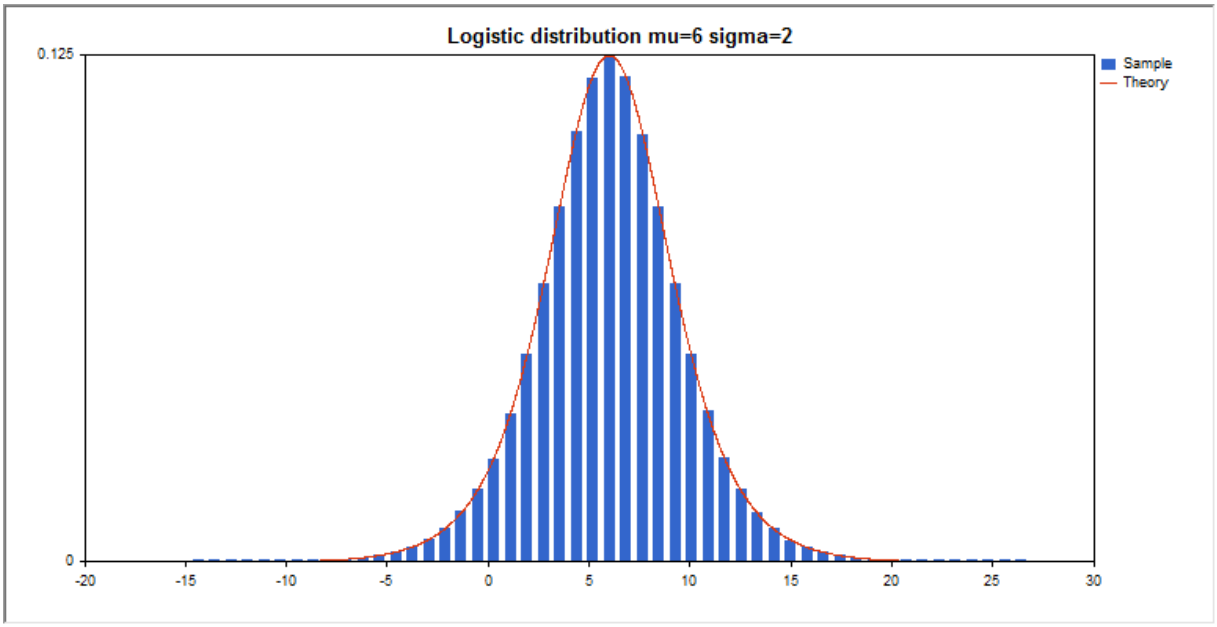
로지스틱 분포

이 섹션에서는 로지스틱 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 로지스틱 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 로지스틱 분포는 다음 공식으로 정의됩니다 :

$$f_{Logistic}(x|\mu,\sigma)=\frac{e^{\frac{x-\mu}{\sigma}}}{\sigma\left(1+e^{\frac{x-\mu}{\sigma}}\right)^2}$$

위치:

- x — 무작위 변수의 값
- μ — 분포의 평균 매개변수
- σ — 분포의 스케일 매개변수



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityLogistic	로지스틱 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionLogistic	로지스틱 확률 분포 함수의 값을 계산
MathQuantileLogistic	지정된 확률에 대한 역 로지스틱 분포 함수의 값을 계산
MathRandomLogistic	로지스틱 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열 생성
MathMomentsLogistic	로지스틱 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

© 2000-2025, MetaQuotes Ltd.

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Logistic.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double mu_par=6;         // 분포의 평균 매개변수
input double sigma_par=2;      // 분포의 스케일 매개변수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // 표본의 값 수
   int ncells=51;          // 히스토그램의 구간 수
   double x[];             // 히스토그램 구간의 중심
   double y[];             // 구간 내에 포함되는 표본의 값 수
   double data[];          // 무작위 값의 표본
   double max,min;         // 표본의 최대값 및 최소값
//--- 로지스틱 분포에서 표본 구함
   MathRandomLogistic(mu_par,sigma_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLogistic(x2,mu_par,sigma_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
```

© 2000-2025, MetaQuotes Ltd.

```
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Logistic distribution mu=%G sigma=%G",mu_par,s
   graphic.BackgroundMainSize(16);
//--- Y 축의 자동 스케일링 비활성화
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//|  시퀀스 생성 값을 계산하기                                        |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityLogistic

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 로지스틱 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // 무작위 변수의 값
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 로지스틱 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // 무작위 변수의 값
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 로지스틱 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dlogis()의 아날로그.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 로지스틱 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

mu

[in] 분포의 평균 매개변수.

sigma

[in] 분포의 스케일 매개변수.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionLogistic

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 확률의 로지스틱 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // 무작위 변수의 값
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 뮤 및 시그마 매개변수를 사용하여 확률의 로지스틱 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // 무작위 변수의 값
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 확률의 로지스틱 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 뮤 및 시그마 매개변수를 사용하여 확률의 로지스틱 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 plogis()의 아날로그.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

mu

[in] 분포의 평균 매개변수.

sigma

[in] 분포의 스케일 매개변수.

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileLogistic

지정된 *확률*에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 로지스틱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileLogistic(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 로지스틱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileLogistic(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 로지스틱 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qlogis()의 아날로그.

```
double  MathQuantileLogistic(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 함수는 뮤 및 시그마 매개변수를 사용하여 역 로지스틱 분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileLogistic(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  mu,             // 분포의 평균 매개변수
   const double  sigma,          // 분포의 스케일 매개변수
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

mu

[in] 분포의 평균 매개변수.

sigma

[in] 분포의 스케일 매개변수.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomLogistic

뮤 및 시그마 매개변수를 사용하여 로지스틱 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomLogistic(
   const double  mu,              // 분포의 평균 매개변수
   const double  sigma,           // 분포의 스케일 매개변수
   int&          error_code       // 오류 코드를 저장할 변수
   );
```

뮤 및 시그마 매개변수를 사용하여 로지스틱 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rlogis()의 아날로그.

```
bool  MathRandomLogistic(
   const double  mu,              // 분포의 평균 매개변수
   const double  sigma,           // 분포의 스케일 매개변수
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

mu

[in] 분포의 평균 매개변수.

sigma

[in] 분포의 스케일 매개변수.

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsLogistic

뮤 및 시그마 매개변수를 사용하여 로지스틱 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsLogistic(
   const double  mu,              // 분포의 평균 매개변수
   const double  sigma,           // 분포의 스케일 매개변수
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드에 대한 변수
   );
```

매개변수

mu

[in] 분포의 평균 매개변수.

sigma

[in] 분포의 스케일 매개변수.

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

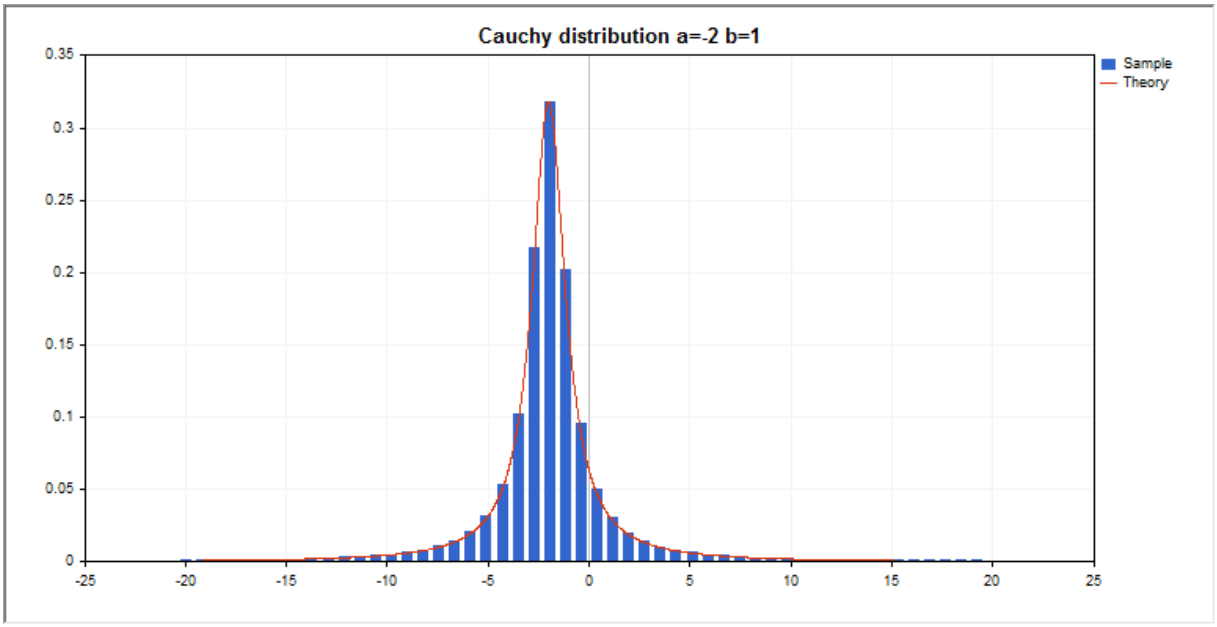
코시 분포

이 섹션에서는 코시(Cauchy) 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산할 수 있고 코시(Cauchy) 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 코시(Cauchy) 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Cauchy}(x|a,b) = \frac{1}{\pi}\frac{b}{\left(b^2+(x-a)^2\right)}$$

위치:

- x — 무작위 변수의 값
- a — 분포의 평균 매개변수
- b — 분포의 스케일 매개변수



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityCauchy	코시 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionCauchy	코시 확률 분포 함수의 값을 계산
MathQuantileCauchy	지정된 확률에 대해 역 코시 분포 함수의 값을 계산
MathRandomCauchy	코시 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsCauchy	코시 분포의 처음 네 포인트에 대한 이론적 수치 값을 계산하기

예:

```
#include <Graphics\Graphic.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Math\Stat\Cauchy.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double a_par=-2;      // 분포의 평균 매개변수
input double b_par=1;       // 분포의 스케일 매개변수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=51;       // 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 코시 분포에서 표본 구함
   MathRandomCauchy(a_par,b_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityCauchy(x2,a_par,b_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("Cauchy distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1)
      return(false);
   int size=ArraySize(data);
   if(size<cells*10)
      return(false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   Print("min=",minv," max=",maxv);
   minv=-20;
   maxv=20;
   double range=maxv-minv;
   double width=range/cells;
   if(width==0)
      return(false);
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=(int)MathRound((data[i]-minv)/width);
      if(ind>=0 && ind<cells)
         frequency[ind]++;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityCauchy

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 코시 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 코시 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 배열 x[]에 대한 a 및 b 매개변수를 사용하여 코시 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dcauchy()의 아날로그.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개 변수를 사용하여 코시 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

a

[in] 분포의 평균 매개변수.

b

[in] 분포의 스케일 매개변수.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionCauchy

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 코시 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 코시 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 코시 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 배열 x[]에 대한 a 및 b 매개변수를 사용하여 코시 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 plogis()의 아날로그.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

a

[in] 분포의 평균 매개변수.

b

[in] 분포의 스케일 매개변수.

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileCauchy

지정된 *확률*에 대해 함수는 a 및 b 매개변수를 사용하여 역 코시 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileCauchy(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 a 및 b 매개변수를 사용하여 역 코시 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileCauchy(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *probability[]* 배열에 대해 함수는 a 및 b 매개변수를 사용하여 역 코시 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qcauschy()의 아날로그.

```
double  MathQuantileCauchy(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

지정된 *probability[]* 배열에 대해 함수는 a 및 b 매개변수를 사용하여 역 코시 분포 함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileCauchy(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

a

[in] 분포의 평균 매개변수.

b

[in] 분포의 스케일 매개변수.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomCauchy

a 및 b 매개변수를 사용하여 코시 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomCauchy(
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

a 및 b 매개변수를 사용하여 코시 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rcauchy()의 아날로그.

```
bool  MathRandomCauchy(
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

a

[in] 분포의 평균 매개변수.

b

[in] 분포의 스케일 매개변수.

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsCauchy

a 및 b 매개변수를 사용하여 코시 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsCauchy(
   const double  a,              // 분포의 평균 매개변수
   const double  b,              // 분포의 스케일 매개변수
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

a

[in] 분포의 평균 매개변수.

b

[in] 분포의 스케일 매개변수.

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

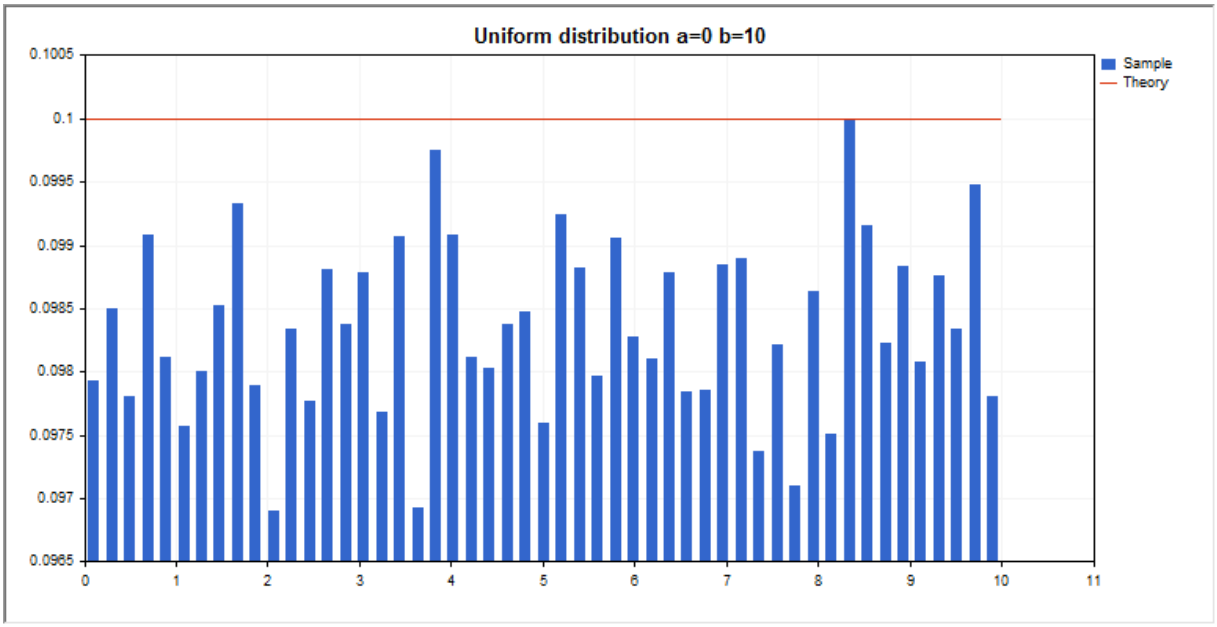
균등 분포

이 섹션에서는 균등 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 균등 법칙에 따라 분포된 의사 난수 생성할 수 있습니다. 균등 분포는 다음 공식으로 정의됩니다:

$$f_{Uniform}(x|\,a,b)=\frac{1}{b-a}$$

위치:

- x —무작위 변수의 값
- a — 분포 매개변수(하한)
- b —분포 매개변수(상한)



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityUniform	균등 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionUniform	균등 확률 분포 함수의 값을 계산합니다
MathQuantileUniform	지정된 확률에 대한 역 균등 분포 함수의 값을 계산합니다
MathRandomUniform	균등 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsUniform	균등 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

```
#include <Graphics\Graphic.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Math\Stat\Uniform.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double a_par=0;      // 분포 매개변수 a(하한)
input double b_par=10;     // 분포 매개변수 b (상한)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // 표본의 값 수
   int ncells=51;          // 히스토그램의 구간 수
   double x[];             // 히스토그램 구간의 중심
   double y[];             // 구간 내에 포함되는 표본의 값 수
   double data[];          // 무작위 값의 표본
   double max,min;         // 표본의 최대값 및 최소값
//--- 균등 분포에서 표본 구함
   MathRandomUniform(a_par,b_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityUniform(x2,a_par,b_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("Uniform distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityUniform

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 균등 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityUniform(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 균등 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityUniform(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 균등 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dunif()의 아날로그.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 균등 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

a

[in] 분포 매개변수 a (하한).

b

[in] 분포 매개변수 b (상한).

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionUniform

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 균등 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 균등 분포의 확률 분포 함수를 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 균등 분포의 확률 분포 함수를 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 균등 분포의 확률 부포 함수를 계산합니다. 오류인 경우 false를 반환합니다. R의 punif()의 아날로그.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

a

[in] 분포 매개변수 a (하한).

b

[in] 분포 매개변수 b (상한).

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileUniform

지정된 *확률*에 대해 함수는 a 및 b 매개변수를 사용하여 역 균등 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileUniform(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 a 및 b 매개변수를 사용하여 역 균등 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileUniform(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수 a 및 b 매개변수를 사용하여 역 균등 분포 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qcauschy()의 아날로그.

```
double  MathQuantileUniform(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수 a 및 b 매개변수를 사용하여 역 균등 분포 함수 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileUniform(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

a

[in] 분포 매개변수 a (하한).

b

[in] 분포 매개변수 b (상한).

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomUniform

a 및 b 매개변수로 균등 분포 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomUniform(
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

a 및 b 매개변수와 함께 균등 분포 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 runif()의 아날로그.

```
bool  MathRandomUniform(
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

a

[in] 분포 매개변수 a (하한).

b

[in] 분포 매개변수 b (상한).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsUniform

a 및 b 매개변수를 사용하여 균등 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsUniform(
   const double  a,              // 분포 매개변수 a (하한)
   const double  b,              // 분포 매개변수 b (상한)
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

a

[in] 분포 매개변수 a (하한).

b

[in] 분포 매개변수 b (상한).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

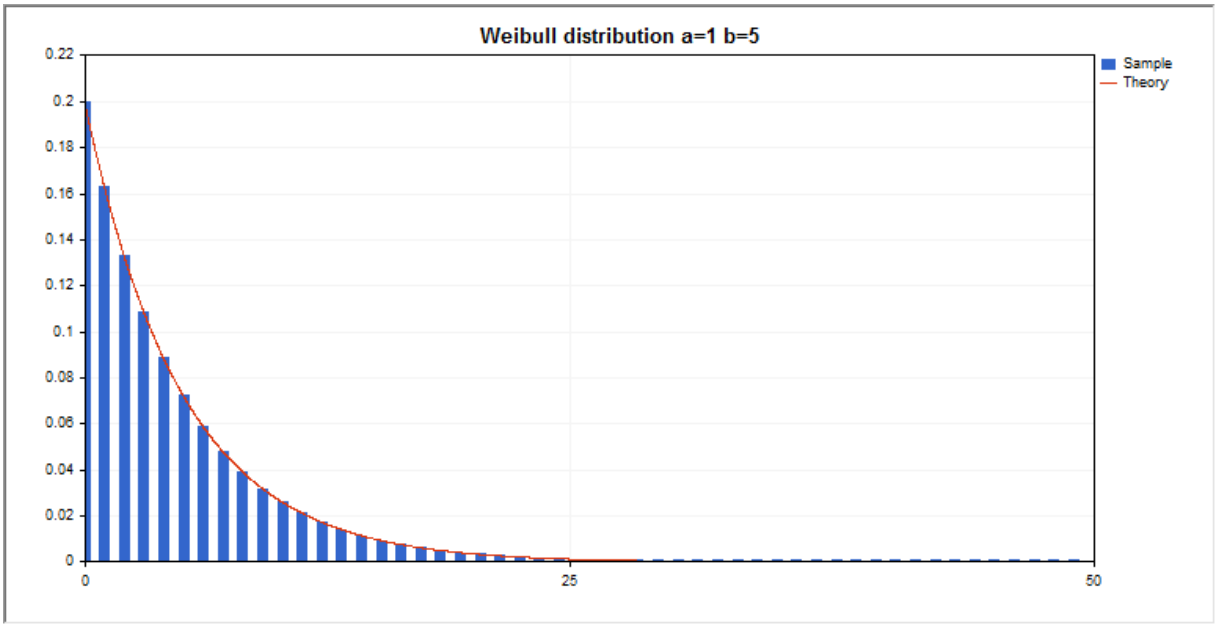
Weibull 분포

이 섹션에는 Weilbull 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산할 수 있고 Weilbull 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. Weilbull 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Weibull}(x|a,b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}}$$

위치:

- x — 무작위 변수의 값
- a — 분포의 매개변수 (형상)
- b — 분포의 매개변수 (스케일)



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityWeibull	Weibull 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionWeibull	Weibull 확률 분포 함수의 값을 계산
MathQuantileWeibull	지정된 확률에 대한 역 Weibull 분포 함수의 값을 계산
MathRandomWeibull	Weibull 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsWeibull	Weibull 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

```
#include <Graphics\Graphic.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Math\Stat\Weibull.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double a_par=1;      // 분포의 매개변수 (형상)
input double b_par=5;      // 분포의 매개변수 (스케일)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=51;       // 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- Weibull 분포에서 표본 구함
   MathRandomWeibull(a_par,b_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityWeibull(x2,a_par,b_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.BackgroundMain(StringFormat("Weibull distribution a=%G b=%G",a_par,b_par));
   graphic.BackgroundMainSize(16);
//--- X 축의 자동 스케일링 비활성화
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
```

© 2000-2025, MetaQuotes Ltd.

```
   {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
    double range=MathAbs(maxv-minv);
    int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
    maxv=NormalizeDouble(maxv,degree);
    minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
    stepv=NormalizeDouble(MathPow(10,-degree),degree);
    if((maxv-minv)/stepv<10)
       stepv/=10.;
   }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityWeibull

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 Weibull 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 Weibull 분포의 확률 밀도 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 Weibull 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dweibull()의 아날로그.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 Weibull 분포의 확률 밀도 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

a

[in] 분포의 매개변수 (스케일).

b

[in] 분포의 매개변수 (형상).

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionWeibull

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 Weibull 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 a 및 b 매개변수를 사용하여 Weibull 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // 무작위 변수의 값
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 Weibull 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 pweibull()의 아날로그.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 a 및 b 매개변수를 사용하여 Weibull 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

a

[in] 분포의 매개변수 (스케일).

b

[in] 분포의 매개변수 (형상).

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileWeibull

지정된 *확률*에 대해 함수는 a 및 b 매개변수를 사용하여 역 Weibull 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileWeibull(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 a 및 b 매개변수를 사용하여 역 Weibull 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileWeibull(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 a 및 b 매개변수를 사용하여 역 Weibull 분포함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qweibull()의 아날로그.

```
double  MathQuantileWeibull(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 a 및 b 매개변수를 사용하여 역 Weibull 분포함수의 값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileWeibull(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

a

[in] 분포의 매개변수 (스케일).

b

[in] 분포의 매개변수 (형상).

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomWeibull

a 및 b 매개변수를 사용하여 Weibull 분포 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomWeibull(
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

a 및 b 매개변수를 사용하여 Weibull 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rweibull()의 아날로그.

```
bool  MathRandomWeibull(
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

a

[in] 분포의 매개변수 (스케일).

b

[in] 분포의 매개변수 (형상).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsWeibull

a 및 b 매개변수를 사용하여 Weibull 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsWeibull(
   const double  a,              // 분포의 매개변수 (형상)
   const double  b,              // 분포의 매개변수 (스케일)
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

a

[in] 분포의 매개변수 (스케일).

b

[in] 분포의 매개변수 (형상).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

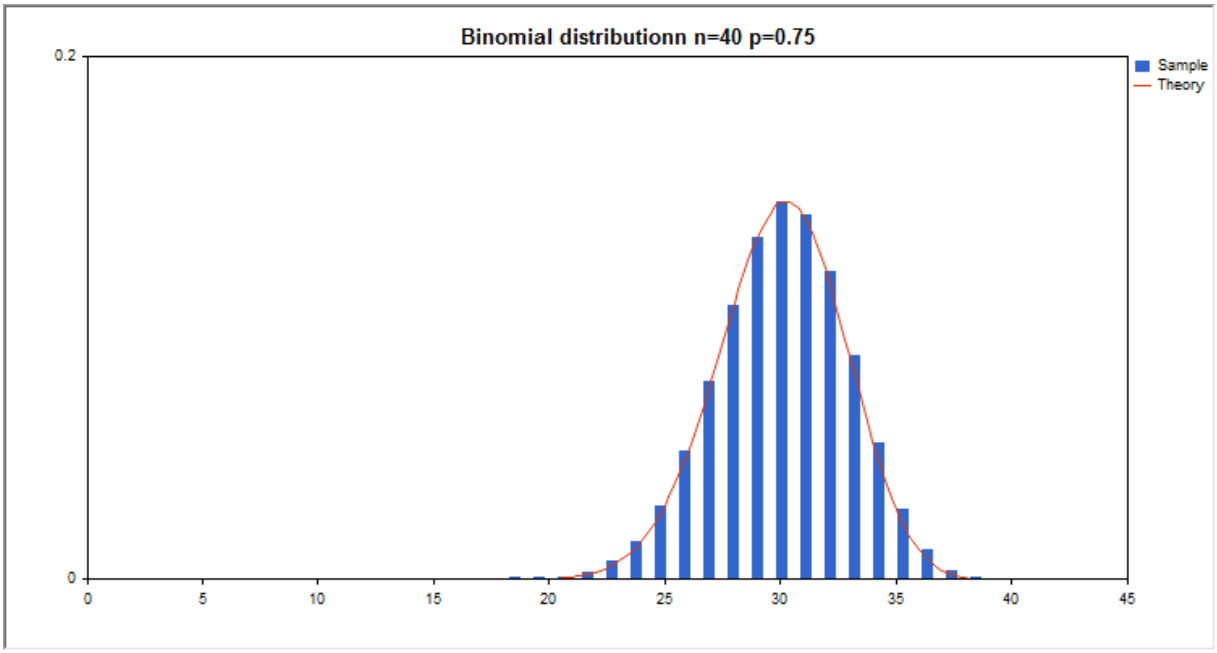
이항 분포

이 섹션에서는 이항 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 이항 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 이항 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Binomial}(x \mid n, p) = \binom{n}{x} p^x (1-p)^{n-x}$$

위치:

- x — 무작위 변수의 값
- n — 테스트 수
- p — 각 테스트의 성공 확률



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityBinomial	이항 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionBinomial	이항 확률 분포 함수의 값을 계산
MathQuantileBinomial	지정된 확률에 대한 역 이항 분포 함수의 값을 계산
MathRandomBinomial	이항 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsBinomial	이항 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Binomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double n_par=40;         // 테스트의 수
input double p_par=0.75;       // 각 테스트의 성공 확률
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // 표본의 값 수
   int ncells=20;          // 히스토그램의 구간 수
   double x[];             // 히스토그램 구간의 중심
   double y[];             // 구간 내에 포함되는 표본의 값 수
   double data[];          // 무작위 값의 표본
   double max,min;         // 표본의 최대값 및 최소값
//--- 이항 분포에서 표본 구함
   MathRandomBinomial(n_par,p_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityBinomial(x2,n_par,p_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Binomial distributionn n=%G p=%G",n_par,p_par)
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityBinomial

무작위 변수 x에 대한 n 및 p 매개변수를 사용하여 이항 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // 무작위 변수의 값
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 n 및 p 매개변수를 사용하여 이항 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // 무작위 변수의 값
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 n 및 p 매개변수를 사용하여 이항 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. Analog of the dbinom() in R.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 n 및 p 매개변수를 사용하여 이항 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

n

[in] 분포의 매개변수(테스트 수).

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionBinomial

랜덤변수 x에 대한 n 및 p 매개변수를 사용하여 이항 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // 무작위 변수의 값
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확률
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

랜덤변수 x에 대한 n 및 p 매개변수를 사용하여 이항 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // 무작위 변수의 값
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 배열에 대한 n 및 p 매개변수를 사용하여 이항 법칙에 대한 확률 분포 함수 x[] 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 pweibull()의 아날로그.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    tail,           // 계산 플래그가 true인 경우 무작위 변수가 x를 초과하지 않
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 지
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 n 및 p 매개변수를 사용하여 이항 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

n

[in] 분포의 매개변수(테스트 수).

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

tail

[in] 계산 플래그. true인 경우 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileBinomial

지정된 *확률*에 대해 함수는 n 및 p 매개변수를 사용하여 이항 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileBinomial(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 n 및 p 매개변수를 사용하여 이항 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileBinomial(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 함수는 n 및 p 매개변수를 사용하여 이항 법칙에 대한 분포 함수의 역값을 계산합니다. 오류인 경우 false를 반환합니다. Analog of the qbinom() in R.

```
double  MathQuantileBinomial(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 함수는 n 및 p 매개변수를 사용하여 이항 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileBinomial(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

n

[in] 분포의 매개변수(테스트 수).

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomBinomial

n 및 p 매개변수를 사용하여 이항 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomBinomial(
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

n 및 p 매개변수를 사용하여 이항 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rweibull()의 아날로그.

```
bool  MathRandomBinomial(
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

n

[in] 분포의 매개변수(테스트 수).

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsBinomial

n 및 p 매개변수를 이용하여 이항 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsBinomial(
   const double  n,              // 분포의 매개변수(테스트 수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

n

[in] 분포의 매개변수(테스트 수).

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

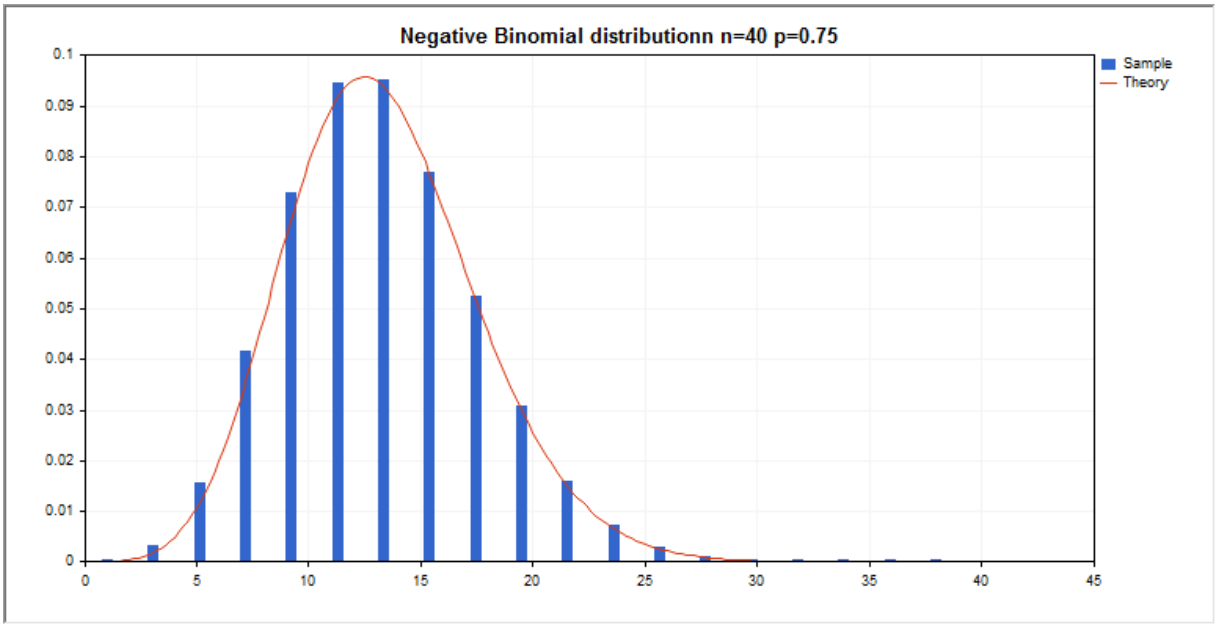
음의 이항 분포

이 섹션에는 음의 이항 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산할 수 있고 음의 이항 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 음의 이항 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{NegativeBinomial}(x \mid r, p) = \frac{\Gamma(r+x)}{\Gamma(r)\Gamma(x+1)} p^r (1-p)^x$$

위치:

- x — 무작위 변수의 값
- r — 테스트 성공 횟수
- p — 성공할 확률



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityNegativeBinomial	음의 이항 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionNegativeBinomial	음의 이항 확률 분포 함수의 값을 계산
MathQuantileNegativeBinomial	지정된 확률에 대한 역 이항 분포 함수의 값을 계산
MathRandomNegativeBinomial	음의 이항 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsNegativeBinomial	음의 이항 분포의 처음 네 모멘트에 대한 이론적 수치값을 계산

예:

© 2000-2025, MetaQuotes Ltd.

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NegativeBinomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double n_par=40;        // 테스트의 수
input double p_par=0.75;      // 각 테스트의 성공 확률
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=19;       // the number of intervals in the histogram
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 음의 이항 분포에서 표본 구함
   MathRandomNegativeBinomial(n_par,p_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityNegativeBinomial(x2,n_par,p_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Negative Binomial distributionn n=%G p=%G",n_p
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityNegativeBinomial

무작위 변수 x에 대한 r 및 p 매개변수를 사용하여 음의 이항 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // 무작위 변수의 값(정수)
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 r 및 p 매개변수를 사용하여 음의 이항 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // 무작위 변수의 값(정수)
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 r 및 p 매개변수를 사용하여 음의 이항 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dnbinom()의 아날로그.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 r 및 p 매개변수를 사용하여 음의 이항 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

r

[in] 테스트 성공 횟수

p

[in] 성공할 확률.

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionNegativeBinomial

변량 변수 x에 대한 r 및 p 매개변수를 사용하여 음의 이항 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // 무작위 변수의 값(정수)
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

변량 변수 x에 대한 r 및 p 매개변수를 사용하여 음의 이항 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // 무작위 변수의 값(정수)
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 r 및 p 매개변수를 사용하여 음의 이항 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 pweibull()의 아날로그.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 r 및 p 매개변수를 사용하여 음의 이항 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수 값이 있는 배열.

r

[in] 테스트 성공 횟수.

p

[in] 성공할 확률.

tail

[in] 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산하기 위한 플래그 log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileNegativeBinomial

지정된 *probability*에 대해 함수는 r 및 p 매개변수를 사용하여 음의 이항 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *probability*에 대해 함수는 r 및 p 매개변수를 사용하여 음의 이항 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 r 및 p 매개변수를 사용하여 음의 이항 법칙에 대한 분포 함수의 역값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qnbinom()의 아날로그.

```
double  MathQuantileNegativeBinomial(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 r 및 p 매개변수를 사용하여 음의 이항 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileNegativeBinomial(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 확률 값을 갖는 배열.

r

[in] 테스트 성공 횟수.

p

[in] 성공할 확률.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomNegativeBinomial

r 및 p 매개변수를 사용하여 음의 이항 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomNegativeBinomial(
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

r 및 p 매개변수를 사용하여 음의 이항 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rweibull()의 아날로그.

```
bool  MathRandomNegativeBinomial(
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

r

[in] 테스트 성공 횟수.

p

[in] 성공할 확률.

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsNegativeBinomial

r 및 p 매개변수를 사용하여 음의 이항 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsNegativeBinomial(
   const double  r,              // 테스트 성공 횟수
   const double  p,              // 성공할 확률
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

r

[in] 테스트 성공 횟수.

p

[in] 성공할 확률.

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

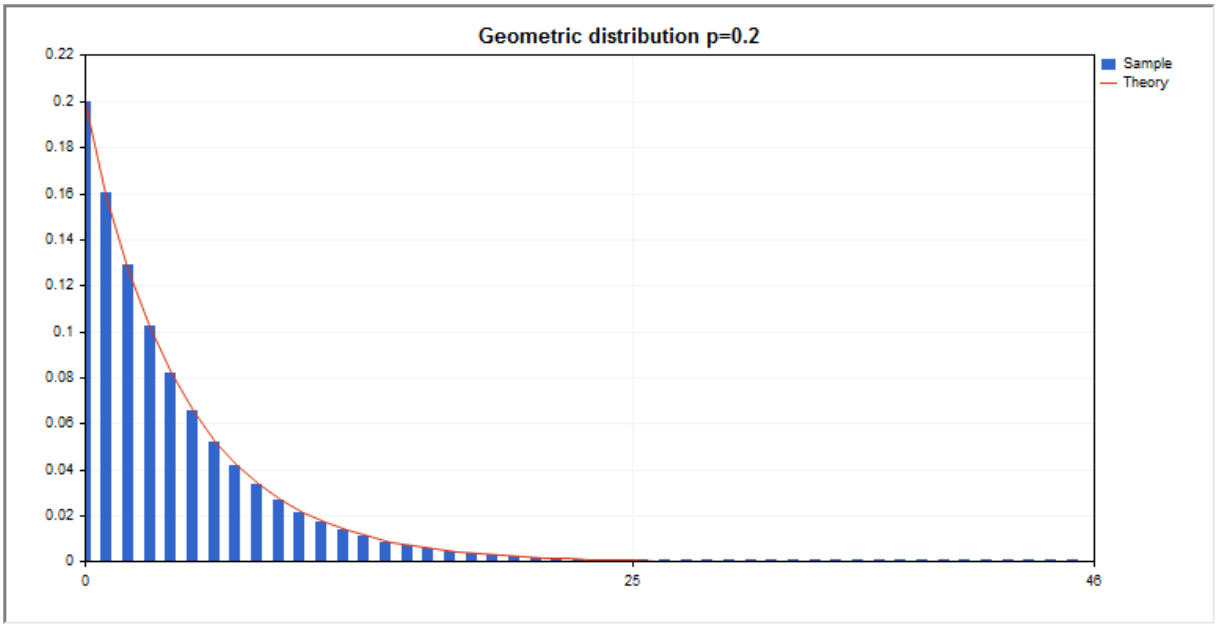
기하 분포

이 절에는 기하 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 기하 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다 기하 분포는 다음 공식으로 정의됩니다:

$$f_{Geometric}(x|p) = p(1-p)^x$$

위치:

- x — 무작위 변수의 값(정수)
- p — 한 테스트에서 이벤트가 발생할 확률



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityGeometric	기하 분포의 확률 밀도 함수를 계산합니다
MathCumulativeDistributionGeometric	기하 확률 분포 함수의 값을 계산
MathQuantileGeometric	지정된 확률에 대한 역 기하 분포 함수의 값을 계산합니다
MathRandomGeometric	기하 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성합니다
MathMomentsGeometric	기하학적 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Geometric.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double p_par=0.2;      // 한 테스트에서 이벤트가 발생할 확률
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=47;       // the 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 기하 분포에서 표본 구함
   MathRandomGeometric(p_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(0,ncells,1,x2);
   MathProbabilityDensityGeometric(x2,p_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Geometric distribution p=%G",p_par));
   graphic.BackgroundMainSize(16);
```

© 2000-2025, MetaQuotes Ltd.

```
//--- X 축의 자동 스케일링 비활성화
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
```

© 2000-2025, MetaQuotes Ltd.

```
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityGeometric

무작위 변수 x에 대한 p 매개변수를 사용하여 기하 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // 무작위 변수의 값(정수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 p 매개변수를 사용하여 기하 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // 무작위 변수의 값(정수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 p 매개변수를 사용하여 기하학적 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dgeom()의 아날로그.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

랜덤 변수 x[] 배열에 대한 p 매개변수를 사용하여 기하 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] 値の ログを 計算する フラグ. log_mode=true面 確率 密度の 自然 ログが 返されます.

error_code

[out] エラー コードを 保存する 変数.

result[]

[out] 確率 密度 関数 値の 配列.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionGeometric

무작위 변수 x에 대한 p 매개변수를 사용하여 기하 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // 무작위 변수의 값(정수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 p 매개변수를 사용하여 기하 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // 무작위 변수의 값(정수)
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 p 매개변수를 사용하여 기하학적 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 pgeom()의 아날로그.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 p 매개변수를 사용하여 기하 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   double&       result[]        // 확률 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 計算 플래그 tail=true이면 랜덤 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode

[in] 값의 로그를 계산하기 위한 플래그 log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileGeometric

지정된 *확률*에 대해 함수는 p 매개변수를 사용하여 기하 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileGeometric(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 p 매개변수를 사용하여 기하 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileGeometric(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 p 매개변수를 사용하여 기하 법칙에 대한 분포 함수의 역값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qgeom()의 아날로그.

```
double  MathQuantileGeometric(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 p 매개변수를 사용하여 기하 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileGeometric(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

[in] 무작위 변수의 확률 값을 갖는 배열.

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomGeometric

p 매개변수를 사용하여 기하 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomGeometric(
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

p 매개변수를 사용하여 기하 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다 오류인 경우 false를 반환합니다. R의 rgeom() 아날로그.

```
bool  MathRandomGeometric(
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsGeometric

p 매개 변수를 사용하여 기하 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsGeometric(
   const double  p,              // 분포의 매개변수(한 테스트에서 사건 발생 확률)
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

p

[in] 분포의 매개변수(한 테스트에서 사건 발생 확률).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

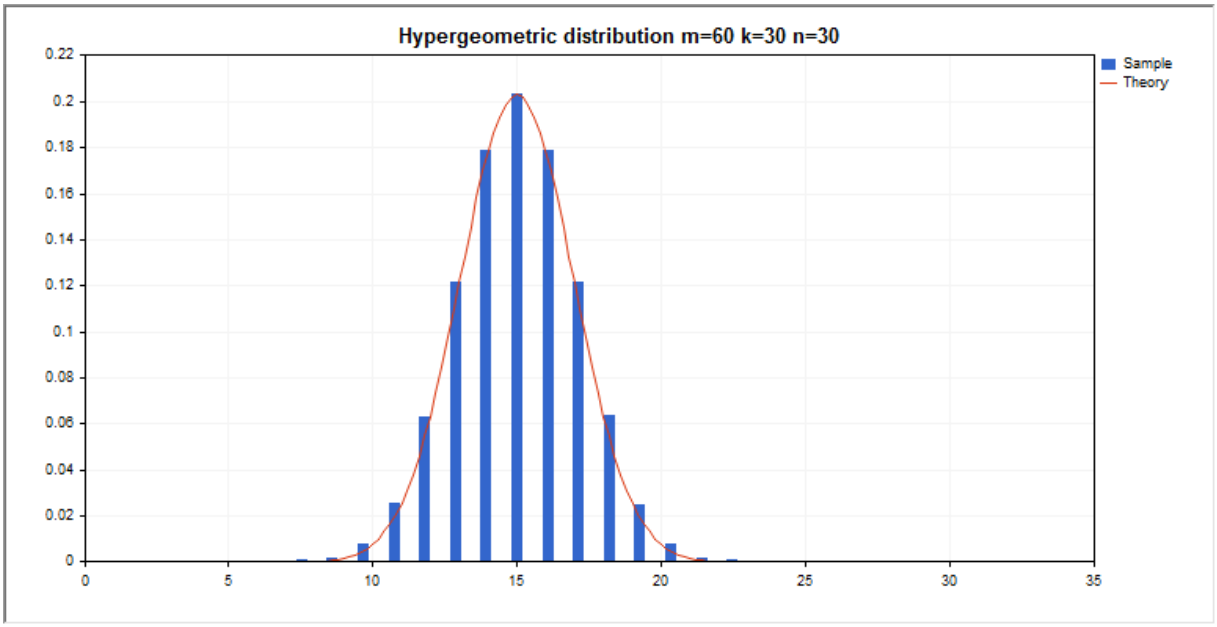
초기하 분포

이 섹션에서는 초기하 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 부위수를 계산하고 초기하 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 초기하 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Hypergeometric}(x \mid m,k,n) = \frac{\binom{k}{x}\binom{m-k}{n-x}}{\binom{m}{n}}$$

위치:

- x — 무작위 변수의 값(정수)
- m — 총 개체 수
- k — 원하는 특성을 가진 개체 수
- n — 개체 그리기 수



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityHypergeometric	초기하 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionHypergeometric	초기하 분포 함수의 값을 계산
MathQuantileHypergeometric	지정된 확률에 대한 역 초기하 분포 함수의 값을 계산
MathRandomHypergeometric	초기하 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성

© 2000-2025, MetaQuotes Ltd.

함수	설명
MathMomentsHypergeometric	초기하 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Hypergeometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- 입력 매개변수
input double m_par=60;      // 총 개체 수
input double k_par=30;      // 원하는 특성을 가진 개체 수
input double n_par=30;      // 개체 그리기 수
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // 표본의 값 수
   int ncells=15;       // the number of intervals in the histogram
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 초기하 분포에서 표본 구함
   MathRandomHypergeometric(m_par,k_par,n_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityHypergeometric(x2,m_par,k_par,n_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
```

© 2000-2025, MetaQuotes Ltd.

```
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Hypergeometric distribution m=%G k=%G n=%G",m_
   graphic.BackgroundMainSize(16);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityHypergeometric

무작위 변수 x에 대한 m,k 및 n 매개변수를 사용하여 초기하 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // 무작위 변수의 값(정수)
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 m,k 및 n 매개변수를 사용하여 초기하 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // 무작위 변수의 값(정수)
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 m,k 및 n 매개변수를 사용하여 초기하 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dhyper()의 아닐로그.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 m,k 및 n 매개변수를 사용하여 초기하 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

© 2000-2025, MetaQuotes Ltd.

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

m

[in] 총 개체 수(정수).

k

[in] 요구되는 특성을 가진 개체 수(정수).

n

[in] 개체 그리기 수(정수).

log_mode

[in] 값의 로그를 계산할 플래그. log_mode=true면 확률 밀도의 자연 로그가 반환됩니다.

error_code

[out] 오류 코드를 저장할 변수.

result[]

[out] 확률 밀도 함수 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionHypergeometric

무작위 변수 x에 대한 m, k 및 n 매개변수를 사용하여 초기하 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // 무작위 변수의 값(정수)
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 m, k 및 n 매개변수를 사용하여 초기하 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // 무작위 변수의 값(정수)
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 m, k 및 n 매개변수를 사용하여 초기하 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dhyper()의 아닐로그.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 분포 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 m,k 및 n 매개변수를 사용하여 초기하 법칙에 대한 확률 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   double&       result[]        // 분포 함수의 값에 대한 배열
   );
```

© 2000-2025, MetaQuotes Ltd.

매개변수

x
[in] 무작위 변수의 값.

x[]
[in] 무작위 변수 값이 있는 배열.

m
[in] 총 개체 수(정수).

k
[in] 요구되는 특성을 가진 개체 수(정수).

n
[in] 개체 그리기 수(정수).

tail
[in] 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode
[in] 값의 로그를 계산하기 위한 플래그 log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code
[out] 오류 코드를 저장할 변수.

result[]
[out] 분포 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantileHypergeometric

지정된 *확률*에 대해 함수는 m,k 및 n 매개변수를 사용하여 초기하 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileHypergeometric(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 m,k 및 n 매개변수를 사용하여 초기하 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantileHypergeometric(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 m, k 및 n 매개변수를 사용하여 초기하 법칙에 대한 분포 함수의 역값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qhyper()의 아날로그.

```
double  MathQuantileHypergeometric(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률 값의 지정된 *probability[]* 배열에 대해 m, k 및 n 매개변수를 사용하여 초기하 법칙에 대한 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantileHypergeometric(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

© 2000-2025, MetaQuotes Ltd.

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

[in] 무작위 변수의 확률 값을 갖는 배열.

m

[in] 총 개체 수(정수).

k

[in] 요구되는 특성을 가진 개체 수(정수).

n

[in] 개체 그리기 수(정수).

tail

[in] 계산플래그 tail=false이면 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomHypergeometric

m, n 및 k 매개변수를 사용하여 초기하 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomHypergeometric(
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

m, n 및 k 매개변수를 사용하여 초기하 분포의 법칙에 따라 분포하는 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rgeom() 아날로그.

```
bool  MathRandomHypergeometric(
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   const int     data_count,     // 필요한 데이터 양
   double&       result[]        // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

m
  [in] 총 개체 수(정수).

k
  [in] 요구되는 특성을 가진 개체 수(정수).

n
  [in] 개체 그리기 수(정수).

error_code
  [out] 오류 코드를 저장할 변수.

data_count
  [out] 필요한 데이터의 양.

result[]
  [out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsHypergeometric

m,n 및 k 매개변수를 사용하여 초기하 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsHypergeometric(
   const double  m,              // 총 개체 수(정수)
   const double  k,              // 요구되는 특성을 가진 개체 수(정수)
   const double  n,              // 개체 그리기 수(정수)
   double&       mean,           // 평균에 대한 변수
   double&       variance,       // 분산에 대한 변수
   double&       skewness,       // 왜도에 대한 변수
   double&       kurtosis,       // 첨도에 대한 변수
   int&          error_code      // 오류 코드에 대한 변수
   );
```

매개변수

m

[in] 총 개체 수(정수).

k

[in] 요구되는 특성을 가진 개체 수(정수).

n

[in] 개체 그리기 수(정수).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

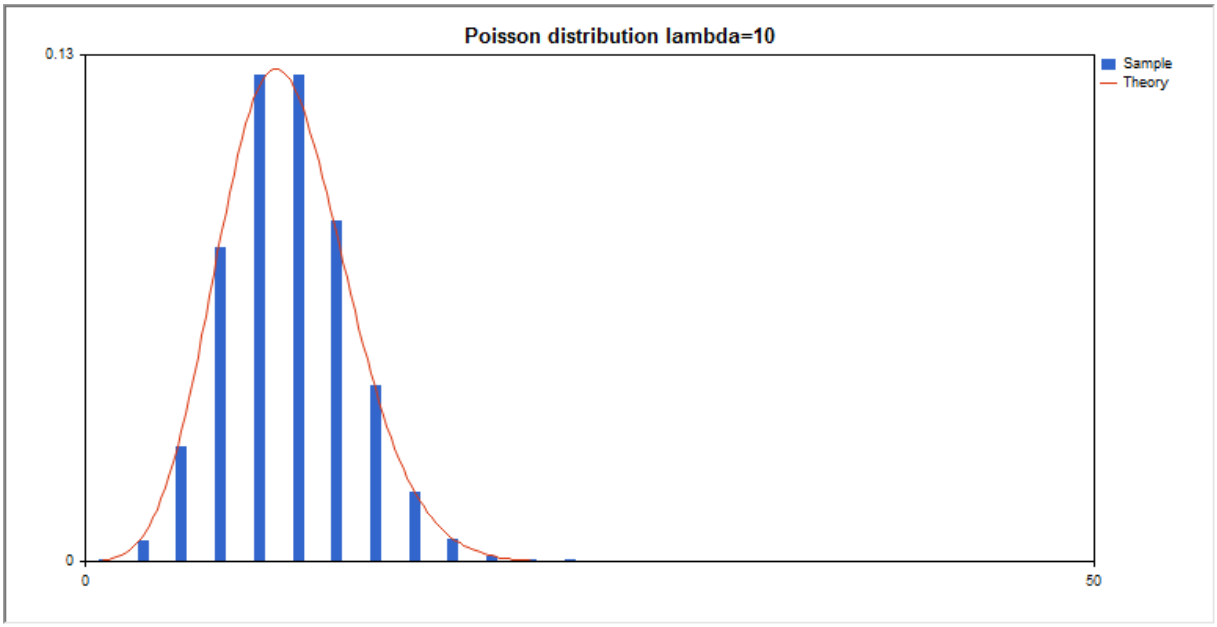
푸와송 분포

이 섹션에는 푸와송 분포로 수행하는 함수가 포함되어 있습니다. 밀도, 확률, 분위수를 계산하고 푸와송 법칙에 따라 분포된 의사 난수를 생성할 수 있습니다. 푸와송 분포는 다음과 같은 공식으로 정의됩니다:

$$f_{Poisson}(x \mid \lambda) = \frac{\lambda^x}{x!} e^{-\lambda}$$

위치:

- x — 무작위 변수의 값
- λ — 분포의 매개변수(평균)



개별 랜덤 변수의 계산 외에도 라이브러리는 랜덤 변수의 배열로 작업할 수 있는 기능도 구현합니다.

함수	설명
MathProbabilityDensityPoisson	푸와송 분포의 확률 밀도 함수를 계산
MathCumulativeDistributionPoisson	푸와송 확률 분포 함수의 값을 계산
MathQuantilePoisson	지정된 확률에 대한 역 푸와송 분포 함수의 값을 계산
MathRandomPoisson	푸와송 법칙에 따라 분포된 의사 난수 변수/의사 난수 변수 배열을 생성
MathMomentsPoisson	푸와송 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산

예:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Poisson.mqh>
#include <Math\Stat\Math.mqh>
```

© 2000-2025, MetaQuotes Ltd.

```
#property script_show_inputs
//--- 입력 매개변수
input double lambda_par=10;      // 분포의 매개변수(평균)
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 가격 차트 숨기기
   ChartSetInteger(0,CHART_SHOW,false);
//--- 난수 생성기 초기화
   MathSrand(GetTickCount());
//--- 무작위 변수 표본 생성
   long chart=0;
   string name="GraphicNormal";
   int n=100000;        // 표본 값의 수
   int ncells=13;       // 히스토그램의 구간 수
   double x[];          // 히스토그램 구간의 중심
   double y[];          // 구간 내에 포함되는 표본의 값 수
   double data[];       // 무작위 값의 표본
   double max,min;      // 표본의 최대값 및 최소값
//--- 푸와송 분포에서 표본 구함
   MathRandomPoisson(lambda_par,n,data);
//--- 히스토그램을 표시할 데이터 계산
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- 순서 경계와 이론적 곡선을 그리는 단계를 구함
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- 이론적으로 계산된 자료를 [min,max]의 간격으로 구함
   double x2[];
   double y2[];
   MathSequence(0,int(MathCeil(max)),1,x2);
   MathProbabilityDensityPoisson(x2,lambda_par,false,y2);
//--- 스케일 설정
   double theor_max=y2[ArrayMaximum(y2)];
   double sample_max=y[ArrayMaximum(y)];
   double k=sample_max/theor_max;
   for(int i=0; i<ncells; i++)
      y[i]/=k;
//--- 출력 차트
  CGraphic 그래픽;
   if(ObjectFind(chart,name)<0)
      graphic.Create(chart,name,0,0,0,780,380);
   else
      graphic.Attach(chart,name);
   graphic.BackgroundMain(StringFormat("Poisson distribution lambda=%G",lambda_par));
   graphic.BackgroundMainSize(16);
//--- Y 축의 자동 스케일링 비활성화
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,2));
   graphic.YAxis().Min(0);
//--- 모든 곡선 플롯
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- 이제 분포 밀도의 이론적 곡선을 플롯합니다
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- 모든 곡선 플롯
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  데이터 집합의 빈도 계산                                         |
//+------------------------------------------------------------------+
bool CalculateHistogramArray(const double &data[],double &intervals[],double &frequenc
                             double &maxv,double &minv,const int cells=10)
  {
   if(cells<=1) return (false);
   int size=ArraySize(data);
   if(size<cells*10) return (false);
   minv=data[ArrayMinimum(data)];
   maxv=data[ArrayMaximum(data)];
   double range=maxv-minv;
   double width=range/cells;
   if(width==0) return false;
   ArrayResize(intervals,cells);
   ArrayResize(frequency,cells);
//--- 구간 중심 정의하기
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- 구간 내에 떨어지는 빈도를 채우기
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  시퀀스 생성 값을 계산하기                                       |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- 순서의 절대 범위를 계산하여 정규화의 정밀도를 구함
   double range=MathAbs(maxv-minv);
```

© 2000-2025, MetaQuotes Ltd.

```
   int degree=(int)MathRound(MathLog10(range));
//--- 최대값과 최소값을 지정된 정밀도로 정규화하기
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- 시퀀스 생성 단계도 지정된 정밀도에 따라 설정됩니다
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityPoisson

무작위 변수 x에 대한 람다 매개변수를 사용하여 푸와송 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // 무작위 변수의 값(정수)
   const double  lambda,         // 분포의 매개변수(평균)
   const bool    log_mode,       // 값의 로그를 계산 log_mode=true이면 확률 밀도의 자연 로그
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 람다 매개변수를 사용하여 푸와송 분포의 확률 질량 함수 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // 무작위 변수의 값(정수)
   const double  lambda,         // 분포의 매개변수(평균)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 람다 매개변수를 사용하여 푸와송 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dhyper()의 아닐로그.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  lambda,         // 분포의 매개변수(평균)
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률 밀도의
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 람다 매개변수를 사용하여 푸와송 분포의 확률 질량 함수 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  lambda,         // 분포의 매개변수(평균)
   double&       result[]        // 확률 밀도 함수 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

lambda

[in] 분포의 매개변수(평균).

log_mode

© 2000-2025, MetaQuotes Ltd.

[in] 値の ログを 計算する フラグ. log_mode=true面 確率 密度の 自然 ログが 返されます.

error_code

[out] エラー コードを 保存する 変数.

result[]

[out] 確率 密度 関数 値の 配列.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionPoisson

무작위 변수 x에 대한 람다 매개변수를 사용하여 푸와송 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // 무작위 변수의 값(정수)
   const double  lambda,         // 분포의 매개변수(평균)
   const bool    tail,           // 계산 플래그 true면 무작위 변수가 x를 초과하지 않을 확률이
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true면 확률의 자연
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x에 대한 람다 매개변수를 사용하여 푸와송 분포 함수의 값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // 무작위 변수의 값(정수)
   const double  lambda,         // 분포의 매개변수(평균)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

무작위 변수 x[] 배열에 대한 람다 매개변수를 사용하여 푸와송 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다. R의 dhyper()의 아닐로그.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  lambda,         // 분포의 매개변수(평균)
   const bool    tail,           // 계산 플래그가 true면 무작위 변수가 x을 초과하지 않을 확
   const bool    log_mode,       // 값의 로그를 계산하는 플래그 log_mode=true이면 확률의 자
   double&       result[]        // 분포 함수의 값에 대한 배열
   );
```

무작위 변수 x[] 배열에 대한 람다 매개면수를 사용하여 푸와송 분포 함수의 값을 계산합니다. 오류인 경우 false를 반환합니다.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // 무작위 변수 값을 갖는 배열
   const double  lambda,         // 분포의 매개변수(평균)
   double&       result[]        // 분포 함수의 값에 대한 배열
   );
```

매개변수

x

[in] 무작위 변수의 값.

x[]

[in] 무작위 변수 값이 있는 배열.

lambda

[in] 분포의 매개변수(평균).

© 2000-2025, MetaQuotes Ltd.

tail
[in] 계산 플래그가 true면 무작위 변수가 x를 초과하지 않을 확률이 계산됩니다.

log_mode
[in] 값의 로그를 계산하기 위한 플래그 log_mode=true이면 확률의 자연 로그가 계산됩니다.

error_code
[out] 오류 코드를 저장할 변수.

result[]
[out] 분포 함수의 값에 대한 배열.

© 2000-2025, MetaQuotes Ltd.

MathQuantilePoisson

지정된 *확률*에 대해 함수는 람다 매개변수를 사용하여 푸와송 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantilePoisson(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  lambda,         // 분포의 매개변수(평균)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

지정된 *확률*에 대해 함수는 람다 매개변수를 사용하여 푸와송 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathQuantilePoisson(
   const double  probability,    // 무작위 변수 발생 확률 값
   const double  lambda,         // 분포의 매개변수(평균)
   int&          error_code      // 오류 코드를 저장할 변수
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 람다 매개변수를 사용하여 푸와송 분포 함수의 역값을 계산합니다. 오류인 경우 false를 반환합니다. R의 qhyper()의 아날로그.

```
double  MathQuantilePoisson(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  lambda,         // 분포의 매개변수(평균)
   const bool    tail,           // 계산 플래그 false인 경우 1.0 확률에 대해 계산이 수행됩니
   const bool    log_mode,       // 계산의 플래그 log_mode=true면 Exp(확률)에 대해 계산이
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

확률값의 지정된 *probability[]* 배열에 대해 함수는 람다 매개변수를 사용하여 푸와송 분포 함수의 역값을 계산합니다. 오류가 발생한 경우 false를 반환합니다.

```
bool  MathQuantilePoisson(
   const double& probability[],  // 무작위 변수의 확률 값을 갖는 배열
   const double  lambda,         // 분포의 매개변수(평균)
   double&       result[]        // 백분위수 값을 갖는 배열
   );
```

매개변수

probability

[in] 무작위 변수의 확률 값.

probability[]

[in] 무작위 변수의 확률 값을 갖는 배열.

lambda

[in] 분포의 매개변수(평균).

© 2000-2025, MetaQuotes Ltd.

tail

[in] 계산플래그 tail=false이면 1.0 확률에 대해 계산이 수행됩니다.

log_mode

[in] log_mode=true면 계산 플래그가 Exp(확률)에 대해 계산이 수행됩니다.

error_code

[out] 오류 코드를 가져올 변수.

result[]

[out] 분위수 값이 있는 배열.

© 2000-2025, MetaQuotes Ltd.

MathRandomPoisson

람다 매개변수 사용하여 푸와송 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류가 발생한 경우 NaN을 반환합니다.

```
double  MathRandomPoisson(
   const double  lambda,          // 분포의 매개변수(평균)
   int&          error_code       // 오류 코드를 저장할 변수
   );
```

람다 매개변수를 사용하여 푸와송 분포의 법칙에 따라 분포된 의사 난수 변수를 생성합니다. 오류인 경우 false를 반환합니다. R의 rgeom() 아날로그.

```
bool  MathRandomPoisson(
   const double  lambda,          // 분포의 매개변수(평균)
   const int     data_count,      // 필요한 데이터 양
   double&       result[]         // 의사 난수 변수 값이 있는 배열
   );
```

매개변수

lambda

[in] 분포의 매개변수(평균).

error_code

[out] 오류 코드를 저장할 변수.

data_count

[out] 필요한 데이터의 양.

result[]

[out] 의사 난수 변수의 값을 얻기 위한 배열.

© 2000-2025, MetaQuotes Ltd.

MathMomentsPoisson

람다 매개변수를 사용하여 푸와송 분포의 처음 네 모멘트에 대한 이론적 수치 값을 계산합니다.

```
double  MathMomentsPoisson(
   const double  lambda,          // 분포의 매개변수(평균)
   double&       mean,            // 평균에 대한 변수
   double&       variance,        // 분산에 대한 변수
   double&       skewness,        // 왜도에 대한 변수
   double&       kurtosis,        // 첨도에 대한 변수
   int&          error_code       // 오류 코드에 대한 변수
   );
```

매개변수

lambda

[in] 분포의 매개변수(평균).

mean

[out] 평균 값을 가져올 변수.

variance

[out] 분산을 가져올 변수.

skewness

[out] 왜도를 가져올 변수.

kurtosis

[out] 첨도를 가져올 변수.

error_code

[out] 오류 코드를 가져올 변수.

값 반환

모멘트 계산에 성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

하위 함수

기본적인 수학적 연산을 수행하는 함수 그룹: 감마 함수, 베타함수, 팩토리얼, 지수, 밑이 다른 로그, 제곱근 등 계산.

개별 숫자 값(실수 및 정수)과 이러한 값의 배열(별도 또는 원래 배열로 결과 출력)을 처리할 수 있는 기능을 제공합니다.

함수	설명
MathRandomNonZero	0.0 ~1.0 범위의 부동 소수점을 가진 난수를 반환합니다.
MathMoments	배열 요소의 처음 네 모멘트(평균, 분산, 왜도, 첨도)를 계산합니다.
MathPowInt	숫자를 지정된 정수 거듭제곱으로 올립니다.
MathFactorial	지정한 정수의 계수를 계산합니다.
MathTrunc	지정한 수 또는 배열 요소의 정수 부분을 계산합니다.
MathRound	숫자 또는 숫자 배열을 지정된 소수 자릿수로 반올림 합니다.
MathArctan2	탄젠트가 [-pi, pi] 범위에서 지정된 두 숫자의 비율과 같은 각도를 계산합니다.
MathGamma	감마 함수의 값을 계산합니다.
MathGammaLog	감마 함수의 로그를 계산합니다.
MathBeta	베타 함수의 값을 계산합니다.
MathBetaLog	베타 함수의 로그를 계산합니다.
MathBetaIncomplete	불완전한 베타 함수의 값을 계산합니다.
MathGammaIncomplete	불완전한 감마 함수의 값을 계산합니다.
MathBinomialCoefficient	이항 계수를 계산합니다.
MathBinomialCoefficientLog	이항 계수의 로그를 계산합니다.
MathHypergeometric2F2	초기하 함수의 값을 계산합니다.
MathSequence	값(첫 번째 요소, 마지막 요소, 시퀀스 단계)을 기반으로 시퀀스를 생성합니다.
MathSequenceByCount	값(첫 번째 요소, 마지막 요소, 시퀀스 요소 수)을 기준으로 시퀀스를 생성합니다.
MathReplicate	값의 반복 시퀀스를 생성합니다.
MathReverse	요소의 역순으로 값 배열을 생성합니다.
MathIdentical	값의 두 배열을 비교하고 모든 요소가 일치하는 경우 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

함수	설명
MathUnique	고유한 값을 가진 배열만 생성합니다.
MathQuickSortAscending	오름차순으로 정렬하는 기능입니다.
MathQuickSortDescending	내림차순으로 정렬하는 함수.
MathQuickSort	정렬 기능.
MathOrder	정렬 후 배열 요소의 순서에 따라 순열을 갖는 배열을 생성합니다.
MathBitwiseNot	배열 요소에 대한 비트 연산의 결과를 계산합니다.
MathBitwiseAnd	배열 요소에 대한 비트별 AND 연산의 결과를 계산합니다.
MathBitwiseOr	배열 요소의 비트별 OR 연산 결과를 계산합니다.
MathBitwiseXor	배열 요소에 대한 비트별 XOR 연산 결과를 계산합니다.
MathBitwiseShiftL	배열 요소에 대한 비트별 SHL 연산의 결과를 계산합니다.
MathBitwiseShiftR	배열 요소에 대한 비트별 SHR 작업의 결과를 계산합니다.
MathCumulativeSum	누적 합계를 사용하여 배열을 생성합니다.
MathCumulativeProduct	누적 곱을 사용하여 배열을 생성합니다.
MathCumulativeMin	누적 최소값이 있는 배열을 생성합니다.
MathCumulativeMax	누적 최대값이 있는 배열을 생성합니다.
MathSin	배열의 요소 sin(x) 함수 값을 계산합니다.
MathCos	배열의 요소 cos(x) 함수 값을 계산합니다.
MathTan	배열 요소에 대한 tan(x) 함수 값을 계산합니다.
MathArcsin	배열 요소에 대한 arcsin(x) 함수 값을 계산합니다.
MathArccos	배열 요소에 대한 arccos(x) 함수 값을 계산합니다.
MathArctan	배열 요소에 대한 arctan(x) 함수 값을 계산합니다.
MathSinPi	배열 요소에 대한 sin(pi*x) 함수 값을 계산합니다.
MathCosPi	배열 요소에 대한 cos(pi*x) 함수 값을 계산합니다.

© 2000-2025, MetaQuotes Ltd.

함수	설명
MathTanPi	배열 요소에 대한 tan(pi*x) 함수 값을 계산합니다.
MathAbs	배열 요소의 절대값을 계산합니다.
MathCeil	배열 요소에 대해 가장 가까운 큰 정수를 반환합니다.
MathFloor	배열 요소에 대해 가장 가까운 작은 정수를 반환합니다.
MathSqrt	배열 요소의 제곱근을 계산합니다.
MathExp	배열 요소에 대한 exp(x) 함수 값을 계산합니다.
MathPow	배열 요소에 대한 pow(x, power) 함수 값을 계산합니다.
MathLog	배열 요소에 대한 log(x) 함수 값을 계산합니다.
MathLog2	배열 요소의 밑이 2인 로그를 계산합니다.
MathLog10	배열 요소의 밑이 10인 로그를 계산합니다.
MathDifference	요소 차이가 y[i]=x[i+lag]-x[i]인 배열을 생성합니다.
MathSample	베열 요소에서 임의 표본을 생성합니다.
MathTukeySummary	배열 요소에 대한 Tukey의 5개 숫자 요약을 계산합니다.
MathRange	배열 요소의 최소값과 최대값을 계산합니다.
MathMin	모든 배열 요소의 최소값을 반환합니다.
MathMax	모든 배열 요소의 최대값을 반환합니다.
MathSum	배열 요소의 합계를 반환합니다.
MathProduct	배열 요소의 곱을 반환합니다.
MathStandardDeviation	배열 요소의 표준 편차를 계산합니다.
MathAverageDeviation	배열 요소의 평균 절대 편차를 계산합니다.
MathMedian	배열 요소의 중위수를 계산합니다.
MathMean	배열 요소의 평균 값을 계산합니다.
MathVariance	배열 요소의 분산 값을 계산합니다.
MathSkewness	배열 요소의 왜도를 계산합니다.
MathKurtosis	배열 요소의 첨도를 계산합니다.
MathLog1p	배열 요소의 log(1+x) 함수 값을 계산합니다.

© 2000-2025, MetaQuotes Ltd.

함수	설명
MathExpm1	배열 요소의 exp(x)-1 함수 값을 계산합니다.
MathSinh	배열 요소의 sinh(x) 함수 값을 계산합니다.
MathCosh	배열 요소의 cosh(x) 함수 값을 계산합니다.
MathTanh	배열 요소의 tanh(x) 함수 값을 계산합니다.
MathArcsinh	배열 요소의 arcsinh(x) 함수 값을 계산합니다.
MathArccosh	배열 요소의 arccosh(x) 함수 값을 계산합니다.
MathArctanh	배열 요소의 arctanh(x) 함수 값을 계산합니다.
MathSignif	값을 matissa의 지정된 자릿수로 반올림 합니다.
MathRank	배열 요소의 랭크를 계산합니다.
MathCorrelationPearson	Pearson의 상관 계수를 계산합니다.
MathCorrelationSpearman	Spearman의 상관 계수를 계산합니다.
MathCorrelationKendall	Kendall의 상관 계수를 계산합니다.
MathQuantile	지정된 확률에 해당하는 표본 분위수를 계산합니다.
MathProbabilityDensityEmpirical	랜덤 값에 대한 경험적 확률 밀도 함수를 계산합니다.
MathCumulativeDistributionEmpirical	랜덤 값에 대한 경험적 누적 분포 함수를 계산합니다.

© 2000-2025, MetaQuotes Ltd.

MathRandomNonZero

0.0～1.0 범위의 부동 소수점을 가진 난수를 반환합니다.

```
double  MathRandomNonZero()
```

값 반환

0.0～1.0 범위의 부동 소수점이 있는 난수입니다.

© 2000-2025, MetaQuotes Ltd.

MathMoments

배열 요소의 처음 네 모멘트(평균, 분산, 왜도, 첨도)를 계산합니다.

```
bool  MathMoments(
   const double&  array[],            // 값의 배열
   double&        mean,               // 평균에 대한 변수
   double&        variance,           // 분산에 대한 변수
   double&        skewness,           // 왜도에 대한 변수
   double&        kurtosis,           // 첨도에 대한 변수
   const int      start=0,            // 초기 인덱스
   const int      count=WHOLE_ARRAY   // 요소의 수
   )
```

매개변수

array[]

[in] 값의 배열.

mean

[out] 평균에 대한 변수(첫 번째 모멘트).

variance

[out] 분산에 대한 변수(두 번째 모멘트).

skewness

[out] 왜도에 대한 변수(세 번째 모멘트).

kurtosis

[out] 첨도에 대한 변수(네 번째 모멘트).

start=0

[in] 계산할 초기 인덱스.

count=WHOLE_ARRAY

[in] 계산할 요소 수.

값 반환

모멘트가 성공적으로 계산되면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathPowInt

숫자를 지정된 정수 거듭제곱으로 올립니다.

```
double  MathPowInt(
   const double  x,      // 수의 값
   const int     power   // 다음을 올릴 파워
   )
```

매개변수

x

[in] 거듭 제곱할 부동 소수점 숫자.

power

[in] 파워를 지정하는 정수.

값 반환

지정된 거듭 제곱으로 상승된 숫자 **x**.

© 2000-2025, MetaQuotes Ltd.

MathFactorial

지정한 정수의 계수를 계산합니다.

```
double  MathFactorial(
   const int  n   // 수의 값
   )
```

매개변수

n

[in] 계산될 팩토리얼 정수.

값 반환

수의 팩토리얼.

© 2000-2025, MetaQuotes Ltd.

MathTrunc

지정한 수 또는 배열 요소의 정수 부분을 계산합니다.

이중 정밀 부동 소수점 번호로 작업하기 위한 버전:

```
double  MathTrunc(
   const double   x          // 수의 값
   )
```

값 반환

지정한 숫자의 정수 부분.

이중 정밀 부동 소수점 번호 배열로 작업하기 위한 버전. 결과가 새 배열로 출력됩니다:

```
bool  MathTrunc(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

이중 정밀 부동 소수점 번호 배열로 작업하기 위한 버전. 결과가 원래 배열로 출력됩니다:

```
bool  MathTrunc(
   double&        array[]    // 값의 배열
   )
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

매개변수

x

[in] 구할 정수 부분인 이중 정밀 부동 소수점 숫자.

array[]

[in] 구할 정수 부분인 이중 정밀 부동 소수점 숫자의 배열.

array[]

[out] 출력 값의 배열.

result[]

[out] 출력 값의 배열.

© 2000-2025, MetaQuotes Ltd.

MathRound

이중 정밀 부동 소숫점 수자 또는 이러한 숫자의 배열을 지정된 소수 자릿수로 반올림합니다.

지정된 소수 자릿수로 이중 정밀 부동 소숫점 숫자를 반올림하기 위한 버전:

```
double  MathRound(
   const double   x,          // 수의 값
   const int      digits      // 소수점 이하 자릿수
   )
```

값 반환

소수 자릿수가 지릿수와 같은 x 매개변수와 가장 가까운 숫자.

이중 정밀 부동 소수점 숫자 배열을 지정된 소수 자릿수로 반올림하기 위한 버전. 결과가 새 배열로 출력됩니다.

```
bool  MathRound(
   const double&  array[],    // 값의 배열
   int            digits,     // 소수점 이하 자릿수
   double&        result[]    // 결과의 배열
   )
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

이중 정밀 부동 소수점 숫자 배열을 지정된 소수 자릿수로 반올림하기 위한 버전. 결과가 원래 배열로 출력됩니다.

```
bool  MathRound(
   double&        array[],    // 값의 배열
   int            digits      // 소수점 이하 자릿수
   )
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

매개변수

x

[in] 반올림할 이중 정밀 부동 소수점 숫자.

digits

[in] 반환된 값의 소수 자릿수.

array[]

[in] 반올림할 이중 정밀 부동 소수점 자릿수 배열.

array[]

[out] 출력 값 배열.

© 2000-2025, MetaQuotes Ltd.

result[]

[out] 출력 값 배열.

© 2000-2025, MetaQuotes Ltd.

MathArctan2

두 인수(x, y)의 몫이 아크탄젠트를 반환합니다.

지정된 두 숫자 (x, y)의 비율로 작업용 버전:

```
double  MathArctan2(
   const double       y,          // Y 좌표
   const double       x           // X 좌표
   )
```

값 반환

(x, y)는 카테시안 좌표계의 점에서 라디안 단위로 측정된 각도 θ는 −π≤θ≤π and tan (θ) = y 또는 x.

x 및 y 배열의 요소 쌍 비율 작업을 위한 버전:

```
bool  MathArctan2(
   const double&      x[],        // x 값의 배열
   const double&      y[],        // y 값의 배열
   double&            result[]    // 결과의 배열
   )
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

매개변수

y

[in] 점의 Y 좌표.

x

[in] 점의 X 좌표.

x[]

[in] 점의 X 좌표 배열.

y[]

[in] 점의 Y 좌표 배열.

result[]

[out] 결과를 출력할 배열

참고

다음 사항에 유의하십시오.

- 제 1사분면의 (x, y)는 다음 값을 반환합니다: 0 < θ < π/2.
- 제 2사분면의 (x, y)는 다음 값을 반환합니다: π/2 < θ≤π.
- 제 3사분면의 (x, y)는 다음 값을 반환합니다: −π < θ <−π/2.
- 제 4사분면의 (x, y)는 다음 값을 반환합니다: −π/2 < θ < 0.

© 2000-2025, MetaQuotes Ltd.

이러한 사분면 외에 있는 점의 반환 값은 아래와 같습니다.

- y가 0 이고 x가 음수가 아니면 θ = 0.
- y가 0 이고 x가 음수이면 θ = π.
- y 가 양수이고 x가 0이면 θ = π/2.
- y가 음수이고 x가 0이면 θ = -π/2.
- y가 0이고 x도 0이면 θ = -π/2.

x 또는 y 매개변수 값이 NaN이거나 x 및 y 매개변수 값이 양의 무한대 또는 음의 무한대 값과 같으면 메서드는 NaN을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathGamma

실수 변수 *x*에 대한 감마 함수의 값을 계산합니다.

```
double  MathGamma(
   const double  x       // 함수의 인수
   )
```

매개변수

x

[in] 함수의 실수 인수.

값 반환

감마 함수의 값.

© 2000-2025, MetaQuotes Ltd.

MathGammaLog

실수 인수 x에 대한 감마 함수의 로그를 계산합니다.

```
double  MathGammaLog(
   const double  x      // 함수의 인수
   )
```

매개변수

x

[in] 함수의 실수 인수.

값 반환

함수의 로그.

© 2000-2025, MetaQuotes Ltd.

MathBeta

실수 인수 a 및 b에 대한 베타 함수의 값을 계산합니다.

```
double  MathBeta(
   const double  a,      // 함수의 첫 번째 인수
   const double  b       // 함수의 두 번째 인수
   )
```

매개변수

a

[in] 함수의 인수.

b

[in] 함수의 b 인수.

값 반환

함수의 값.

© 2000-2025, MetaQuotes Ltd.

MathBetaLog

실수 인수 a와 b에 대한 베타 함수의 로그를 계산합니다.

```
double  MathBetaLog(
   const double  a,      // 함수의 첫 번째 인수
   const double  b       // 함수의 두 번째 인수
   )
```

매개변수

a

[in] 함수의 인수.

b

[in] 함수의 b 인수.

값 반환

함수의 로그.

© 2000-2025, MetaQuotes Ltd.

MathBetaIncomplete

불완전한 베타 함수의 값을 계산합니다.

```
double  MathBetaIncomplete(
   const double  x,      // 함수의 인수
   const double  p,      // 함수의 첫 번째 매개변수
   const double  q       // 함수의 두 번째 매개변수
   )
```

매개변수

x

[in] 함수의 인수.

p

[in] 베타 함수의 첫 번째 매개변수는 0.0보다 커야 합니다.

q

[in] 베타 함수의 두 번째 매개변수는 0.0보다 커야 합니다.

값 반환

함수의 값.

© 2000-2025, MetaQuotes Ltd.

MathGammaIncomplete

불완전 감마 함수의 값을 계산합니다.

```
double  MathGammaIncomplete(
   double  x,          // 함수의 인수
   double  alpha       // 함수의 매개변수
   )
```

매개변수

x

[in] 함수의 인수.

alpha

[in] 불완전 감마 함수의 매개변수.

값 반환

함수의 값.

© 2000-2025, MetaQuotes Ltd.

MathBinomialCoefficient

다음의 이항 계수를 계산합니다: C(n,k)=n!/(k!*(n-k)!).

```
long  MathBinomialCoefficient(
   const int  n,      // 요소의 총 수
   const int  k       // 컴비네이션의 요소 수
   )
```

매개변수

n

[in] 요소 수.

k

[in] 각 컴비네이션의 요소 수.

값 반환

N 컴비네이션 수가 K를 선택합니다.

© 2000-2025, MetaQuotes Ltd.

MathBinomialCoefficientLog

다음의 이항 계수의 로그를 계산합니다: Log(C(n,k))=Log(n!/(k!*(n-k)!))

정수 인수의 버전:

```
double  MathBinomialCoefficientLog(
   const int      n,       // 요소의 총 수
   const int      k        // 컴비네이션의 요소 수
   )
```

실수 인수의 버전:

```
double  MathBinomialCoefficientLog(
   const double   n,       // 요소의 총 수
   const double   k        // 컴비네이션의 요소 수
   )
```

매개변수

n

[in] 요소 수.

k

[in] 각 컴비네이션의 요소 수.

값 반환

C(n,k)의 로그.

© 2000-2025, MetaQuotes Ltd.

MathHypergeometric2F2

테일러 메서드를 사용하여 Hypergeometric_2F2 (a, b, c, d, z) 함수 값을 계산합니다.

```
double  MathHypergeometric2F2(
   const double  a,      // 함수의 첫 번째 매개변수
   const double  b,      // 함수의 두 번째 매개변수
   const double  c,      // 함수의 세 번째 매개변수
   const double  d,      // 함수의 네 번째 매개변수
   const double  z       // 함수의 다섯 번째 매개변수
   )
```

매개변수

a

[in] 함수의 첫 번째 매개변수.

b

[in] 함수의 두 번째 매개변수.

c

[in] 함수의 세 번째 매개변수.

d

[in] 함수의 네 번째 매개변수.

z

[in] 함수의 다섯 번째 매개변수.

값 반환

함수의 값.

© 2000-2025, MetaQuotes Ltd.

MathSequence

시퀀스의 첫 번째 요소, 마지막 요소, 단계 값을 기준으로 일련의 값을 생성합니다.

실수 값으로 작업할 버전:

```
bool  MathSequence(
   const double  from,        // 초기값
   const double  to,          // 최종값
   const double  step,        // 단계
   double&       result[]     // 결과의 배열
   )
```

정수 값 작업용 버전:

```
bool  MathSequence(
   const int     from,        // 초기값
   const int     to,          // 최종값
   const int     step,        // 단계
   int&          result[]     // 결과의 배열
   )
```

매개변수

from

[in] 시퀀스의 첫 번째 값

to

[in] 시퀀스의 마지막 값

step

[in] 시퀀스의 단계.

result[]

[out] 시퀀스를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathSequenceByCount

첫 번째 요소, 마지막 요소, 시퀀스의 요소 수에 따라 값의 시퀀스를 생성합니다.

실수 값으로 작업할 버전:

```
bool  MathSequenceByCount(
   const double  from,        // 초기값
   const double  to,          // 최종값
   const int     count,       // 숫자
   double&       result[]     // 결과의 배열
   )
```

정수 값 작업용 버전:

```
bool  MathSequenceByCount(
   const int     from,        // 초기값
   const int     to,          // 최종값
   const int     count,       // 숫자
   int&          result[]     // 결과의 배열
   )
```

매개변수

from

[in] 시퀀스의 첫 번째 값.

to

[in] 시퀀스의 마지막 값.

count

[in] 시퀀스의 요소 수.

result[]

[out] 시퀀스를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathReplicate

値の反復シーケンスを生成します。

実数値で作業するバージョン:

```
bool  MathReplicate(
   const double&  array[],    // 値の配列
   const int      count,      // 反復回数
   double&        result[]    // 結果の配列
   )
```

整数値で作業するバージョン:

```
bool  MathReplicate(
   const int&     array[],    // 値の配列
   const int      count,      // 反復回数
   int&           result[]    // 結果の配列
   )
```

パラメータ

array[]

[in] シーケンス生成用の配列。

count

[in] シーケンスで配列を反復する回数。

result[]

[out] 出力シーケンスの配列。

戻り値

成功した場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

MathReverse

요소의 역순으로 값 배열을 생성합니다.

실수 값과 결과 출력을 새 배열로 작업하기 위한 버전:

```
bool  MathReverse(
   const double&  array[],      // 값의 배열
   double&        result[]      // 결과의 배열
   )
```

정수 값 및 결과 출력을 새 배열로 작업하기 위한 버전:

```
bool  MathReverse(
   const int&     array[],      // 값의 배열
   int&           result[]      // 결과의 배열
   )
```

실수 값 및 원본 배열에 대한 결과 출력을 사용하기 위한 버전.

```
bool  MathReverse(
   double&        array[]       // 값의 배열
   )
```

정수 값 및 결과 출력을 원래 배열로 작업하기 위한 버전:

```
bool  MathReverse(
   int&           array[]       // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

array[]

[out] 값의 역순을 가진 출력 배열.

result[]

[out] 값의 역순을 가진 출력 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathIdentical

값의 두 배열을 비교하고 모든 요소가 일치하는 경우 true를 반환합니다.

실수 값 배열 작업용 버전:

```
bool  MathIdentical(
   const double&  array1[],      // 값의 첫 번째 배열
   const double&  array2[]       // 값의 두 번째 배열
   )
```

정수 값 배열로 작업용 버전:

```
bool  MathIdentical(
   const int&     array1[],      // 값의 첫 번째 배열
   const int&     array2[]       // 값의 두 번째 배열
   )
```

매개변수

array1[]
  [in] 비교할 첫 번째 배열.

array2[]
  [in] 비교할 두 번째 배열.

값 반환

배열이 같으면 true를 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathUnique

고유한 값을 가진 배열만 생성합니다.

실수 값으로 작업할 버전:

```
bool  MathUnique(
   const double&   array[],   // 값의 배열
   double&         result[]   // 결과의 배열
   )
```

정수 값 작업용 버전:

```
bool  MathUnique(
   const int&     array[],      // 값의 배열
   int&           result[]      // 결과의 배열
   )
```

매개변수

array[]

[in] 원본 배열.

result[]

[out] 고유 값을 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathQuickSortAscending

QuickSort 알고리즘을 사용하여 array[] 및 indices[] 배열의 동시 오름차순 정렬을 위한 함수입니다.

```
void  MathQuickSortAscending(
   double&  array[],     // 값의 배열
   int&     indices[],   // 인덱스의 배열
   int      first,       // 초기값
   int      last         // 최종값
   )
```

매개변수

array[]

[in][out] 정렬할 배열.

indices[]

[in][out] 원래 배열의 인덱스를 저장할 배열.

first

[in] 정렬을 시작할 요소의 인덱스.

last

[in] 정렬을 중지할 요소의 인덱스.

© 2000-2025, MetaQuotes Ltd.

MathQuickSortDescending

QuickSort 알고리즘을 사용하여 array[] 및 indices[] 배열의 동시 내림차순 정렬을 위한 함수입니다.

```
void  MathQuickSortDescending(
   double&  array[],     // 값의 배열
   int&     indices[],   // 인덱스의 배열
   int      first,       // 초기값
   int      last         // 최종값
   )
```

매개변수

array[]

[in][out] 정렬할 배열.

indices[]

[in][out] 원래 배열의 인덱스를 저장할 배열.

first

[in] 정렬을 시작할 요소의 인덱스.

last

[in] 정렬을 중지할 요소의 인덱스.

© 2000-2025, MetaQuotes Ltd.

MathQuickSort

QuickSort 알고리즘을 사용하여 array[] 및 indices[] 배열을 동시에 정렬하는 함수.

```
void  MathQuickSort(
   double&  array[],     // 값의 배열
   int&     indices[],   // 인덱스의 배열
   int      first,       // 초기값
   int      last,        // 최종값
   int      mode         // 방향
   )
```

매개변수

array[]

[in][out] 정렬할 배열.

indices[]

[in][out] 원래 배열의 인덱스를 저장할 배열.

first

[in] 정렬을 시작할 요소의 인덱스.

last

[in] 정렬을 중지할 요소의 인덱스.

mode

[in] 정렬 방향 (0보다 크면 상승, 그렇지 않으면 하락).

© 2000-2025, MetaQuotes Ltd.

MathOrder

정렬 후 배열 요소의 순서에 따라 순열을 갖는 정수 배열을 생성합니다.

실수 값 배열로 작업하기 위한 버전:

```
bool  MathOrder(
   const double&  array[],   // 값의 배열
   int&           result[]   // 결과의 배열
   )
```

정수 값 배열로 작업하기 위한 버전:

```
bool  MathOrder(
   const int&     array[],   // 값의 배열
   int&           result[]   // 결과의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 정렬된 인덱스를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseNot

배열 요소에 대한 비트 연산의 결과를 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathBitwiseNot(
   const int&  array[],   // 값의 배열
   int&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathBitwiseNot(
   int&        array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

array[]
  [out] 출력 값의 배열.

result[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseAnd

지정된 배열에 대한 비트별 AND 연산 결과를 계산합니다.

```
bool  MathBitwiseAnd(
   const int&  array1[],  // 값의 첫 번째 배열
   const int&  array2[],  // 값의 두 번째 배열
   int&        result[]   // 결과의 배열
   )
```

매개변수

array1[]

[in] 값의 첫 번째 배열.

array2[]

[in] 값의 두 번째 배열.

result[]

[out] 결과를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseOr

지정된 배열에 대한 비트별 OR 연산의 결과를 계산합니다.

```
bool  MathBitwiseOr(
   const int&  array1[],  // 값의 첫 번째 배열
   const int&  array2[],  // 값의 두 번째 배열
   int&        result[]   // 결과의 배열
   )
```

매개변수

array1[]

[in] 값의 첫 번째 배열.

array2[]

[in] 값의 두 번째 배열.

result[]

[out] 결과를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseXor

지정된 배열에 대한 비트별 XOR 작업의 결과를 계산합니다.

```
bool  MathBitwiseXor(
   const int&  array1[],  // 값의 첫 번째 배열
   const int&  array2[],  // 값의 두 번째 배열
   int&        result[]   // 결과의 배열
   )
```

매개변수

array1[]

[in] 값의 첫 번째 배열.

array2[]

[in] 값의 두 번째 배열.

result[]

[out] 결과를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseShiftL

배열 요소에 대한 비트별 SHL(비트별 왼쪽 이동) 작업의 결과를 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathBitwiseShiftL(
   const int&  array[],    // 값의 배열
   const int   n,          // 값 이동
   int&        result[]    // 결과 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathBitwiseShiftL(
   int&        array[],    // 값의 배열
   const int   n           // 값 이동
   )
```

매개변수

array[]
  [in] 값의 배열.

n
  [in] 이동할 비트 수.

array[]
  [out] 출력 값의 배열.

result[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathBitwiseShiftR

배열 요소에 대한 비트별 SHR(비트별 오른쪽 이동) 작업의 결과를 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathBitwiseShiftR(
   const int&  array[],    // 값의 배열
   const int   n,          // 값 이동
   int&        result[]    // 결과 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathBitwiseShiftR(
   int&        array[],    // 값의 배열
   const int   n           // 값 이동
   )
```

매개변수

array[]
  [in] 값의 배열.

n
  [in] 이동할 비트 수.

array[]
  [out] 출력 값의 배열.

result[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeSum

누적 합계를 사용하여 배열을 생성합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathCumulativeSum(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathCumulativeSum(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]
[in] 값의 배열.

array[]
[out] 출력 값의 배열.

result[]
[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeProduct

누적 곱을 사용하여 배열을 생성합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathCumulativeProduct(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathCumulativeProduct(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeMin

누적 최소값이 있는 배열을 생성합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathCumulativeMin(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathCumulativeMin(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeMax

누적 최대값이 있는 배열을 생성합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathCumulativeMax(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathCumulativeMax(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathSin

배열의 요소 sin(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathSin(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathSin(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCos

배열의 요소 cos(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathCos(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathCos(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathTan

배열 요소에 대한 tan(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathTan(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathTan(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathArcsin

배열 요소에 대한 arcsin(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathArcsin(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathArcsin(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathArccos

배열 요소에 대한 arccos(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathArccos(
   const double&   array[],    // 값의 배열
   double&         result[]    // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathArccos(
   double&         array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathArctan

배열 요소에 대한 arctan(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathArctan(
   const double&   array[],   // 값의 배열
   double&         result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathArctan(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathSinPi

배열 요소에 대한 sin(pi*x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathSinPi(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathSinPi(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCosPi

배열 요소에 대한 cos(pi*x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathCosPi(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathCosPi(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathTanPi

배열 요소에 대한 tan(pi*x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathTanPi(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathTanPi(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathAbs

배열 요소의 절대값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathAbs(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathAbs(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCeil

배열 요소에 대해 가장 가까운 큰 정수를 반환합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathCeil(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathCeil(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathFloor

배열 요소에 대해 가장 가까운 작은 정수를 반환합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathFloor(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathFloor(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathSqrt

배열 요소의 제곱근을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathSqrt(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathSqrt(
   double&         array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathExp

배열 요소에 대한 exp(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathExp(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathExp(
   double&       array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathPow

배열 요소에 대한 pow(x, power) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathPow(
   const double&  array[],    // 값의 배열
   const double   power,      // 파워
   double&        result[]    // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathPow(
   double&        array[],    // 값의 배열
   const double   power       // 파워
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathLog

배열 요소에 대한 log(x) 함수 값을 계산합니다.

결과를 새 배열로 출력하여 자연 로그를 계산용 버전.

```
bool  MathLog(
   const double&  array[],      // 값의 배열
   double&        result[]      // 결과의 배열
   )
```

결과를 원래 배열로 출력하기 위해 자연 로그를 계산하기 위한 버전.

```
bool  MathLog(
   double&        array[]       // 값의 배열
   )
```

결과를 새 배열로 출력하여 지정된 기준으로 로그를 계산하기 위한 버전.

```
bool  MathLog(
   const double&   array[],     // 값의 배열
   const double    base,        // 로그의 밑
   double&         result[]     // 결과의 배열
   )
```

결과를 원래 배열로 출력하여 지정된 기준으로 로그를 계산하는 버전.

```
bool  MathLog(
   double&         array[],     // 값의 배열
   const double    base         // 로그의 밑
   )
```

매개변수

array[]
  [in] 값의 배열.

base
  [in] 로그의 밑.

array[]
  [out] 출력 값의 배열.

result[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathLog2

배열 요소의 밑이 2인 로그를 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathLog2(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathLog2(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathLog10

배열 요소의 밑이 10인 로그를 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathLog10(
   const double&   array[],   // 값의 배열
   double&         result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathLog10(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathLog1p

배열 요소의 log(1+x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathLog1p(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathLog1p(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathDifference

요소 차이가 y[i]=x[i+lag]-x[i]인 배열을 생성합니다.

실수 값 배열의 단일 세대용 버전:

```
bool  MathDifference(
   const double  &array[],     // 값의 배열
   const int     lag,          // 지연
   double        &result[]     // 결과의 배열
   )
```

정수 값 배열의 단일 세대용 버전:

```
bool  MathDifference(
   const int     &array[],     // 값의 배열
   const int     lag,          // 지연
   int           &result[]     // 결과의 배열
   )
```

실수 값 배열의 반복 생성용 버전(입력 매개변수에 반복 횟수가 설정됨):

```
bool  MathDifference(
   const double  &array[],     // 값의 배열
   const int     lag,          // 지연
   const int     differences,  // 반복 횟수
   double        &result[]     // 결과의 배열
   )
```

정수 값 배열의 반복 생성용 버전(입력 매개 변수에 반복 횟수가 설정됨):

```
bool  MathDifference(
   const int&    array[],      // 값의 배열
   const int     lag,          // 지연
   const int     differences,  // 반복 횟수
   int&          result[]      // 결과의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

lag
  [in] 지연 매개변수.

differences
  [in] 반복 횟수.

result[]
  [out] 결과를 출력할 배열.

값 반환

© 2000-2025, MetaQuotes Ltd.

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathSample

베열 요소에서 임의 표본을 생성합니다.

실수 값 배열로 작업하기 위한 버전:

```
bool  MathSample(
   const double&  array[],          // 값의 배열
   const int      count,            // 숫자
   double&        result[]          // 결과의 배열
   )
```

정수 값 배열로 작업하기 위한 버전:

```
bool  MathSample(
   const int&     array[],          // 값의 배열
   const int      count,            // 숫자
   int&           result[]          // 결과의 배열
   )
```

실수 값 배열로 작업하기 위한 버전. 표본을 대체할 수 있습니다:

```
bool  MathSample(
   const double&  array[],          // 값의 배열
   const int      count,            // 숫자
   const bool     replace,          // 플래그
   double&        result[],         // 결과의 배열
   )
```

정수 값 배열로 작업하기 위한 버전. 표본을 대체할 수 있습니다:

```
bool  MathSample(
   const int&     array[],          // 값의 배열
   const int      count,            // 숫자
   const bool     replace,          // 플래그
   int&           result[]          // 결과의 배열
   )
```

표본 추출 확률이 정의되는 실수 값 내열로 작업하기 위한 버전.

```
bool  MathSample(
   const double&  array[],          // 값의 배열
   double&        probabilities[],  // 확률의 배열
   const int      count,            // 숫자
   double&        result[]          // 결과의 배열
   )
```

표본 추출 확률이 정의되는 정수 값의 배열로 작업하기 위한 버전.

```
bool  MathSample(
   const int&     array[],          // 값의 배열
   double&        probabilities[],  // 확률의 배열
```

© 2000-2025, MetaQuotes Ltd.

```
   const int      count,            // 숫자
   int&           result[]          // 결과의 배열
   )
```

표본 추출 확률이 정의되는 실수 값 내열로 작업하기 위한 버전. 표본을 대체할 수 있습니다:

```
bool  MathSample(
   const double&  array[],          // 값의 배열
   double&        probabilities[],  // 확률의 배열
   const int      count,            // 숫자
   const bool     replace,          // 플래그
   double&        result[]          // 결과의 배열
   )
```

표본 추출 확률이 정의되는 정수 값의 배열로 작업하기 위한 버전. 표본을 대체할 수 있습니다:

```
bool  MathSample(
   const int&     array[],          // 값의 배열
   double&        probabilities[],  // 확률의 배열
   const int      count,            // 숫자
   const bool     replace,          // 플래그
   int&           result[]          // 결과의 배열
   )
```

매개변수

array[]

[in] 정수 값의 배열.

probabilities[]

[in] 요소 표본을 추출하기 위한 확률 배열.

count

[in] 요소의 수.

replace

[in] 대체 표본을 허용하는 매개변수.

result[]

[out] 결과를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

참고

replace=true 인수를 사용하면 원래 시퀀스로 복귀하여 요소를 임의로 표본 추출할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

MathTukeySummary

배열 요소에 대한 Tukey의 5개 숫자 요약(최소,하한 사분위, 중위수, 상한 사분위, 최대값)을 계산합니다.

```
bool  MathTukeySummary(
   const double&  array[],       // 값의 배열
   const bool     removeNAN,     // 플래그
   double&        minimum,       // 최소값
   double&        lower_hinge,   // 하한 사분위수
   double&        median,        // 중위수
   double&        upper_hinge,   // 상한 사분위수
   double&        maximum        // 최대값
   )
```

매개변수

array[]

[in] 실수값 배열.

removeNAN

[in] 숫자가 아닌 값을 제거할지 여부를 나타내는 플래그.

minimum

[out] 최소값을 저장하기 위한 변수.

lower_hinge

[out] 하한 사분위수를 저장할 변수.

median

[out] 중위값을 저장할 변수.

upper_hinge

[out] 상한 사분위수를 저장할 변수.

maximum

[out] 최대값을 저장하기 위한 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathRange

배열 요소의 최소값과 최대값을 계산합니다.

```
bool  MathRange(
   const double&  array[],  // 값의 배열
   double&        min,      // 최소값
   double&        max       // 최대값
   )
```

매개변수

array[]

[in] 값의 배열.

min

[out] 최소값을 저장하기 위한 변수.

max

[out] 최대값을 저장하기 위한 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathMin

모든 배열 요소의 최소값을 반환합니다.

```
double  MathMin(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

최소값.

© 2000-2025, MetaQuotes Ltd.

MathMax

모든 배열 요소의 최대값을 반환합니다.

```
double  MathMax(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

최대값.

© 2000-2025, MetaQuotes Ltd.

MathSum

배열 요소의 합계를 반환합니다.

```
double  MathSum(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

요소의 합계.

© 2000-2025, MetaQuotes Ltd.

MathProduct

배열 요소의 곱을 반환합니다.

```
double  MathProduct(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

요소의 곱.

© 2000-2025, MetaQuotes Ltd.

MathStandardDeviation

배열 요소의 표준 편차를 계산합니다.

```
double  MathStandardDeviation(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

표준 편차.

© 2000-2025, MetaQuotes Ltd.

MathAverageDeviation

함수는 배열 요소의 평균 절대 편차를 계산합니다.

```
double  MathAverageDeviation(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

배열 요소의 평균 절대 편차.

© 2000-2025, MetaQuotes Ltd.

MathMedian

배열 요소의 중위수를 계산합니다.

```
double  MathMedian(
   double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

중위값.

© 2000-2025, MetaQuotes Ltd.

MathMean

배열 요소의 평균 값을 계산합니다.

```
double  MathMean(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

평균값.

© 2000-2025, MetaQuotes Ltd.

MathVariance

함수는 배열 요소의 분산(두 번째 모멘트)을 계산합니다.

```
double  MathVariance(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

분산의 값.

© 2000-2025, MetaQuotes Ltd.

MathSkewness

함수는 배열 요소의 왜도(세 번째 모멘트)를 계산.

```
double  MathSkewness(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

왜도.

© 2000-2025, MetaQuotes Ltd.

MathKurtosis

함수는 배열 요소의 첨도(네 번째 모멘트)를 계산합니다.

```
double  MathKurtosis(
   const double&  array[]   // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

값 반환

첨도.

© 2000-2025, MetaQuotes Ltd.

MathExpm1

배열 요소의 exp(x)-1 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathExpm1(
   const double&   array[],   // 값의 배열
   double&         result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathExpm1(
   double&         array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathSinh

배열 요소의 sinh(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathSinh(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathSinh(
   double&       array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCosh

배열 요소의 cosh(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathCosh(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathCosh(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathTanh

배열 요소의 tanh(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathTanh(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathTanh(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathArcsinh

배열 요소의 arcsinh(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathArcsinh(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathArcsinh(
   double&         array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathArccosh

배열 요소의 arccosh(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathArccosh(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathArccosh(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]
  [in] 값의 배열.

result[]
  [out] 출력 값의 배열.

array[]
  [out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathArctanh

배열 요소의 arctanh(x) 함수 값을 계산합니다.

결과가 새 배열로 출력되는 버전:

```
bool  MathArctanh(
   const double&  array[],   // 값의 배열
   double&        result[]   // 결과의 배열
   )
```

결과가 원래 배열로 출력되는 버전:

```
bool  MathArctanh(
   double&        array[]    // 값의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

result[]

[out] 출력 값의 배열.

array[]

[out] 출력 값의 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathSignif

값을 matissa의 지정된 자릿수로 반올림 합니다.

실수 값으로 작업하기 위한 버전:

```
double  MathSignif(
   const double   x,          // 값
   const int      digits      // 소수 자릿수
   )
```

값 반환

반올림 값.

실수 값 배열 및 결과 출력을 별도의 배열로 작업하기 위한 버전:

```
bool  MathSignif(
   const double&  array[],    // 값의 배열
   int            digits,     // 소수 자릿수
   double         result[]    // 결과의 배열
   )
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

실수 값 배열 및 원본 배열로 결과 출력 작업용 버전:

```
bool  MathSignif(
   double&        array[],    // 값의 배열
   int            digits      // 소수 자릿수
   )
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

매개변수

x

[in] 반올림할 실수 값.

digits

[in] 소수 자릿수.

array[]

[in] 실수 값 배열.

array[]

[out] 출력 값의 배열.

result[]

[out] 출력 값의 배열.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

MathRank

배열 요소의 랭크를 계산합니다.

실수 값 배열로 작업하기 위한 버전:

```
bool  MathRank(
   const double&  array[],  // 값의 배열
   double&        rank[]    // 순위의 배열
   )
```

정수 값 배열로 작업하기 위한 버전:

```
bool  MathRank(
   const int&     array[],  // 값의 배열
   double&        rank[]    // 순위의 배열
   )
```

매개변수

array[]

[in] 값의 배열.

rank[]

[out] 랭크를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCorrelationPearson

Pearsonの相関係数を計算します。

実数値の配列で作業するためのバージョン：

```
bool  MathCorrelationPearson(
   const double&  array1[],  // 値の最初の配列
   const double&  array2[],  // 値の2番目の配列
   double&        r          // 相関係数
   )
```

整数値の配列で作業するためのバージョン：

```
bool  MathCorrelationPearson(
   const int&     array1[],  // 値の最初の配列
   const int&     array2[],  // 値の2番目の配列
   double&        r          // 相関係数
   )
```

引数

array1[]

[in] 値の最初の配列。

array2[]

[in] 値の2番目の配列。

r

[out] 相関係数を保存する変数。

戻り値

成功した場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

MathCorrelationSpearman

Spearmanの相関係数を計算します。

実数値配列で作業するバージョン:

```
bool  MathCorrelationSpearman(
   const double&  array1[],  // 値の最初の配列
   const double&  array2[],  // 値の2番目の配列
   double&        r          // 相関係数
   )
```

整数値配列で作業するバージョン:

```
bool  MathCorrelationSpearman(
   const int&     array1[],   // 値の最初の配列
   const int&     array2[],   // 値の2番目の配列
   double&        r           // 相関係数
   )
```

パラメータ

array1[]

[in] 値の最初の配列。

array2[]

[in] 値の2番目の配列。

r

[out] 相関係数を保存する変数。

戻り値

成功した場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

MathCorrelationKendall

Kendallの相関係数を計算します。

実数値配列で作業するバージョン:

```
bool  MathCorrelationKendall(
   const double&  array1[],  // 値の最初の配列
   const double&  array2[],  // 値の2番目の配列
   double&        tau        // 相関係数
   )
```

整数値配列で作業するバージョン:

```
bool  MathCorrelationKendall(
   const int&     array1[],  // 値の最初の配列
   const int&     array2[],  // 値の2番目の配列
   double&        tau        // 相関係数
   )
```

引数

array1[]

[in] 値の最初の配列。

array2[]

[in] 値の2番目の配列。

tau

[out] 相関係数を保存する変数。

戻り値

成功した場合はtrue、それ以外の場合はfalseを返します。

© 2000-2025, MetaQuotes Ltd.

MathQuantile

지정된 확률에 해당하는 표본 분위수를 계산합니다: Q[i](p) = (1 - gamma)*x[j] + gamma*x[j+1]

```
bool  MathQuantile(
   const double&  array[],     // 값의 배열
   const double&  probs[],     // 확률의 배열
   double&        quantile[]   // 분위수를 출력할 배열
   )
```

매개변수

array[]

[in] 값의 배열.

probs[]

[in] 확률의 배열.

quantile[]

[out] 분위수를 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathProbabilityDensityEmpirical

함수는 배열에서 무작위 값에 대한 경험적 확률 밀도 함수(pdf)를 계산합니다.

```
bool  MathProbabilityDensityEmpirical(
   const double&  array[],   // 무작위 값의 배열
   const int      count,     // 쌍의 수
   double&        x[],       // x 값 배열
   double&        pdf[]      // pdf 값의 배열
   )
```

매개변수

array[]

[in] 무작위 값의 배열.

count

[in] (x, pdf(x)) 쌍의 숫자.

x[]

[out] x 값을 출력할 배열.

pdf[]

[out] pdf(x) 값을 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MathCumulativeDistributionEmpirical

함수는 배열에서 무작위 값에 대한 경험적 누적분포함수(cdf)를 계산합니다.

```
bool  MathCumulativeDistributionEmpirical(
   const double&  array[],   // 무작위 값의 배열
   const int      count,     // 쌍의 수
   double&        x[],       // x 값 배열
   double&        cdf[]      // cdf 값 배열
   )
```

매개변수

array[]

[in] 무작위 값의 배열.

count

[in] (x, cdf(x)) 쌍의 수.

x[]

[out] x 값을 출력할 배열.

cdf[]

[out] cdf(x) 값을 출력할 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

퍼지는 퍼지 논리로 작업하기 위한 라이브러리입니다

퍼지 논리는 사실이 언어적 변수로 표시될 때 전통적인 아리스토텔리스 논리학을 종합한 것입니다. 고전적 논리와 동등한 퍼지 논리는 정의된 퍼지 집합에 대한 자체 퍼지 논리 연산을 가지고 있습니다. 피지 집합뿐만 아니라 일반 집합에도 동일한 연산이 있지만, 계산만 훨씬 어렵습니다. 우리는 퍼지 집합의 구성이 퍼지 집합으로 구성된 다는 것도 주목해야 합니다.

퍼지 논린의 주요 원리는 고전 논리와는 별개로 현실의 반영에 대한 최대 근접성과 높은 수준의 주관성이며, 이는 계산에 상당한 오류를 초래할 수 있습니다.

퍼지 모델(또는 시스템)은 퍼지 논리에 기반한 계산의 수학적 모델입니다. 이러한 모델의 구성은 연구 대상이 형식화가 약학 정확한 수학적 설명이 너무 복잡하거나 알려지지 않은 경우에 적용할 수 있습니다. 이러한 모델의 출력 값(오류 모델)의 품질은 이 모델을 설정한 엑스퍼트 어드바이저에만 직접 좌우됩니다. 오류를 최소화하는 최선의 옵션은 가장 완전하고 포괄적인 모델을 그린 다음 대규모 트레이닝 집합에서 머신 러닝으로 조정하는 것입니다.

모델 제작의 진행은 크게 세 단계로 나눌 수 있습니다:

1. 모델의 입력 및 출력 특성에 대한 정의.
2. 기술 기반 구축.
3. 퍼지 추론 방법 중 하나 선택 (Mamdani 또는 Sugeno).

첫 번째 단계는 결과 2에 직접 영향을 미치고 모델의 향후 작동을 결정합니다. 기술 기반 또는 규칙 기반은 검사된 개체의 입력과 출력 사이의 관계를 정의하는 "if,then" 퍼지 규칙 유형의 집합입니다. 시스템 내 규칙 수는 제한되지 않으며 엑스퍼트 어드바이저가 결정하기도 합니다. 퍼지 규칙의 일반화된 형식은 다음과 같습니다:

규칙 조건이면 규칙 결론.

규칙 조건은 개체의 현재 상태와 규칙 결론(이 조건이 개체에 미치는 영향)을 설명합니다. 조건과 결론에 대한 일반적인 견해는 퍼지 추론에 의해 결정되기 때문에 선택할 수 없습니다.

시스템의 각 규칙에는 가중치가 있으며, 이 특성은 모델에서 규칙의 중요성을 정의합니다. 가중치 인자는 [0,1] 범위 내의 규칙에 할당됩니다. 관련 문헌에서 찾을 수 있는 퍼지 모델의 많은 예에서 가중치 데이터는 지정되지 않지만, 그것이 존재하지 않는다는 것을 의미하지는 않습니다. 실제로 데이터베이스의 각 규칙에 대해 가중치는 고정되며 통합과 같습니다. 각 규칙에는 두 가지 유형의 항과 결론이 있을 수 있습니다:

1. 단순 —단일 퍼지 변수를 포함;
2. 복합 —여러 퍼지 변수를 포함.

생성된 지식 기반에 따라 모델에 대한 퍼지 추론 시스템이 결정됩니다. 퍼지 논리적 추론은 지식 기반 및 퍼지 연산을 사용하여 입력의 현재 값에 해당하는 퍼지 집합의 형태로 결론을 얻는 것입니다. 퍼지 추론의 두 가지 주요 유형은 Mamdani 및 Sugeno입니다.

© 2000-2025, MetaQuotes Ltd.

멤버십 함수

멤버십 함수는 퍼지 집합에 대한 범용 집합의 임의의 요소의 멤버십 정도를 계산할 수 있는 함수이다. 결과적으로 멤버십 함수의 도메인은 [0, 1] 범위 내에 있어야 합니다.

대부분의 경우 멤버십 함수는 연속적이고 단조롭습니다.

멤버십 함수의 클래스	설명
CConstantMembershipFunction	좌표축과 병렬로 멤버십 함수를 직선으로 구현하기 위한 클래스
CCompositeMembershipFunction	멤버십 함수 구성을 구현하기 위한 클래스
CDifferencTwoSigmoidalMembershipFunction	A1, A2, C1 및 C2 매개변수를 갖는 두 시그모이드 함수 간의 차이의 형태로 멤버십 함수를 구현하기 위한 클래스
CGeneralizedBellShapedMembershipFunction	A, B 및 C 매개변수로 일반화된 종모양의 멤버십 함수를 구현하기 위한 클래스
CNormalCombinationMembershipFunction	B1, B2, Sigma1 및 Sigma2 매개변수를 사용하여 양방향 가우스 멤버십 함수를 구현하기 위한 클래스
CNormalMembershipFunction	B 및 Sigma 매개변수를 사용하여 대칭 가우스 멤버십 함수를 구현하기 위한 클래스
CP_ShapedMembershipFunction	A, B, C 및 D 매개변수를 사용하여 파이 모양의 멤버십 함수를 구현하기 위한 클래스
CProductTwoSigmoidalMembershipFunction	멤버십 함수를 A1, A2, C1 및 C2 매개변수를 갖는 두 시그모이드 함수의 곱 형태로 구현하기 위한 클래스
CS_ShapedMembershipFunction	A 및 B 매개변수를 사용하여 S와 같은 멤버십 함수를 구현하기 위한 클래스
CTrapezoidMembershipFunction	X1, X2, X3 및 X4 매개변수로 사다리꼴 멤버십 함수를 구현하기 위한 클래스
CTriangularMembershipFunction	X1, X2 및 X3 매개변수를 사용하여 삼각 멤버십 함수를 구현하는 클래스
CSigmoidalMembershipFunction	A 및 C 매개변수를 사용하여 시그모이드 멤버십 함수를 구현하는 클래스
CZ_ShapedMembershipFunction	A 및 B 매개변수를 사용하여 z와 같은 멤버십 함수를 구현하기 위한 클래스.
IMembershipFunction	모든 멤버십 함수 클래스의 기본 클래스.

© 2000-2025, MetaQuotes Ltd.

CConstantMembershipFunction

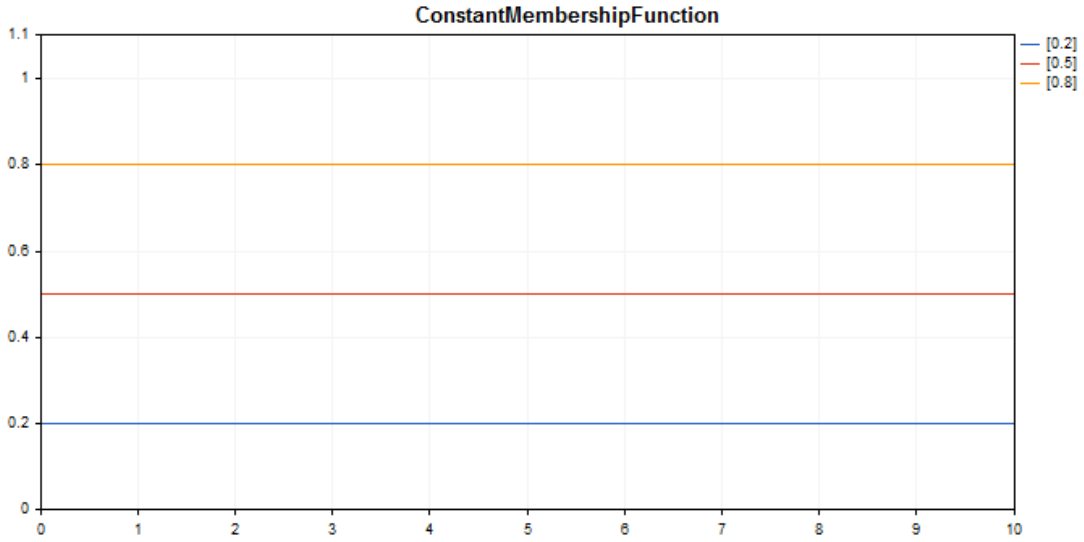
좌표 축과 평행한 직선으로 멤버십 함수를 구현하기 위한 클래스.

설명

함수는 다음과 같은 방정식으로 설명할 수 있습니다:

y(x)=c

따라서 함수의 멤버십 등급은 전체 숫자 축을 따라 동일하며 생성자에 지정된 매개변수와 같습니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CConstantMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
IMembershipFunction
CConstantMembershipFunction

클래스 메서드

클래스 메서드	설명
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                                  ConstantMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CConstantMembershipFunction func1(0.2);
CConstantMembershipFunction func2(0.5);
CConstantMembershipFunction func3(0.8);
//--- 멤버십 함수에 대한 래퍼 생성
double ConstantMembershipFunction1(double x) { return(func1.GetValue(x)); }
double ConstantMembershipFunction2(double x) { return(func2.GetValue(x)); }
double ConstantMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"ConstantMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"ConstantMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ConstantMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(ConstantMembershipFunction1,0.0,10.0,1.0,CURVE_LINES,"[0.2]");
   graphic.CurveAdd(ConstantMembershipFunction2,0.0,10.0,1.0,CURVE_LINES,"[0.5]");
   graphic.CurveAdd(ConstantMembershipFunction3,0.0,10.0,1.0,CURVE_LINES,"[0.8]");
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const double  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

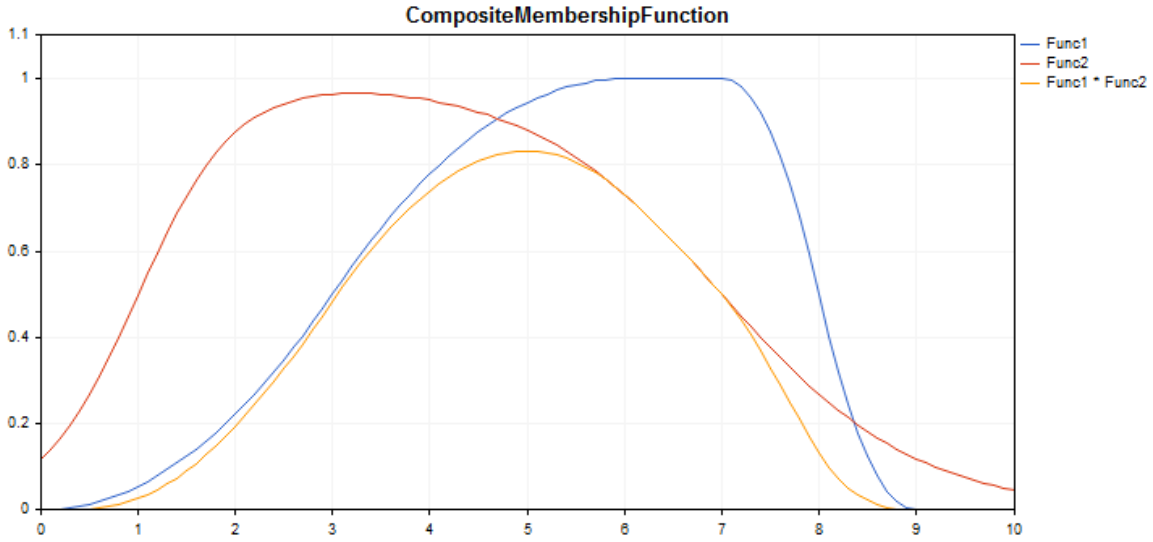
© 2000-2025, MetaQuotes Ltd.

CCompositeMembershipFunction

멤버십 함수의 구성을 구현하기 위한 클래스.

설명

멤버십 함수의 구성은 지정된 연산자를 사용하는 둘 이상의 멤버십 함수의 구성.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CCompositeMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
IMembershipFunction
CCompositeMembershipFunction

클래스 메서드

클래스 메서드	설명
CompositionType	합성 연산자를 설정합니다.
MembershipFunctions	멤버십 함수 목록을 가져옵니다.
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                                  CompositeMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CP_ShapedMembershipFunction func2(0,6,7,9);
CCompositeMembershipFunction composite(ProdMF,GetPointer(func1),GetPointer(func2));
//--- 멤버십 함수에 대한 래퍼 생성
double ProductTwoSigmoidalMembershipFunctions(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction(double x) { return(func2.GetValue(x)); }
double CompositeMembershipFunction(double x) { return(composite.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"CompositeMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"CompositeMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("CompositeMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(P_ShapedMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1");
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions,0.0,10.0,0.1,CURVE_LINES,"F
   graphic.CurveAdd(CompositeMembershipFunction,0.0,10.0,0.1,CURVE_LINES,"Func1 * Func
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.Update();
  }
```

CompositionType

합성 연산자를 설정합니다.

```
void  CompositionType(
   MfCompositionType  value      // 연산자 유형
   )
```

매개변수

값

[in] 구성 연산자 유형.

참고

다음과 같은 연산자 유형을 사용할 수 있습니다:

- MinMF (minimum of functions)
- MaxMF (maximum of functions)
- ProdMF (product of functions)
- SumMF (sum of functions)

MembershipFunctions

구성에 포함된 멤버십 함수 목록을 가져옵니다.

```
CList*  MembershipFunctions(
   void        // 멤버십 함수 목록
   )
```

값 반환

멤버십 함수 목록.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

© 2000-2025, MetaQuotes Ltd.

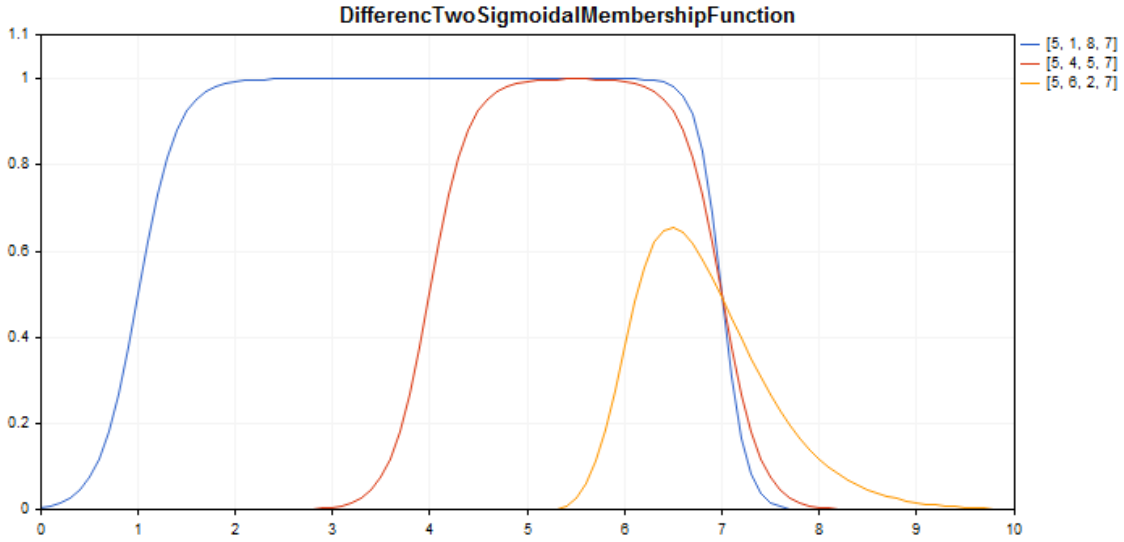
© 2000-2025, MetaQuotes Ltd.

CDifferencTwoSigmoidalMembershipFunction

A1, A2, C1 및 C2 매개변수를 가진 두 시그모이드 함수 간의 차이 형태로 멤버십 함수를 구현하기 위한 등급.

설명

함수는 시그모이드 곡선에 기초합니다. 인수 값으로 시작하는 값이 1인 멤버십 함수를 생성할 수 있습니다. 이러한 기능은 "short" 또는 "long"과 같은 용어를 설정해야 할 경우 적합합니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CDifferencTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
  IMembershipFunction
    CDifferencTwoSigmoidalMembershipFunction

클래스 메서드

클래스 메서드	설명
A1	첫 번째 멤버십 함수 기울기 비율을 가져오고 설정합니다.
A2	두 번째 멤버십 함수 기울기 비율을 가져와 설정하기.

© 2000-2025, MetaQuotes Ltd.

클래스 메서드	설명
C1	첫 번째 멤버십 함수 변곡 좌표 매개변수를 가져와 설정하기.
C2	두 번째 멤버십 함수 변곡 좌표 매개변수를 가져와 설정하기.
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                      DifferencTwoSigmoidalMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd.     |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CDifferencTwoSigmoidalMembershipFunction func1(5,1,8,7);
CDifferencTwoSigmoidalMembershipFunction func2(5,4,5,7);
CDifferencTwoSigmoidalMembershipFunction func3(5,6,2,7);
//--- 멤버십 함수에 대한 래퍼 생성
double DifferencTwoSigmoidalMembershipFunction1(double x) { return(func1.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction2(double x) { return(func2.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"DifferencTwoSigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"DifferencTwoSigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("DifferencTwoSigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(DifferencTwoSigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- X 축 속성 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (Get method)

첫 번째 멤버십 함수 기울기 비율을 가져오기.

```
double  A1()
```

값 반환

기울기 비율 값.

A1 (Set method)

첫 번째 멤버십 함수 기울기 비율 설정하기.

```
void  A1(
   const double  a1      // 기울기 비율 값
   )
```

매개변수

a1

[in] 기울기 비율 값.

A2 (Get method)

두 번째 멤버십 함수 기울기 비율 가져오기.

```
double  A2()
```

값 반환

기울기 비율 값.

A2 (Set method)

두 번째 멤버십 함수 기울기 비율 설정하기.

```
void  A2(
   const double  a2      // 기울기 비율 값
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

매개변수

a2

[in] 기울기 비율 값.

C1 (Get method)

첫 번째 멤버십 함수 변곡 좌표 매개변수 가져오기.

```
double  C1()
```

값 반환

변곡 좌표 값.

C1 (Set method)

첫 번째 멤버십 함수 변곡 좌표 매개변수 설정하기.

```
void  C1(
   const double  c1      // 변곡 좌표 값
   )
```

매개변수

c1

[in] 변곡 좌표 값.

C2 (Get method)

두 번째 멤버십 함수 변곡 좌표 매개변수 가져오기.

```
double  C2()
```

값 반환

변곡 좌표 값.

C2 (Set method)

두 번째 멤버십 함수 변곡 좌표 매개변수 설정하기.

```
void  C2(
   const double  c2      // 변곡 좌표 값
   )
```

매개변수

c2

[in] 변곡 좌표 값.

© 2000-2025, MetaQuotes Ltd.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const double  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

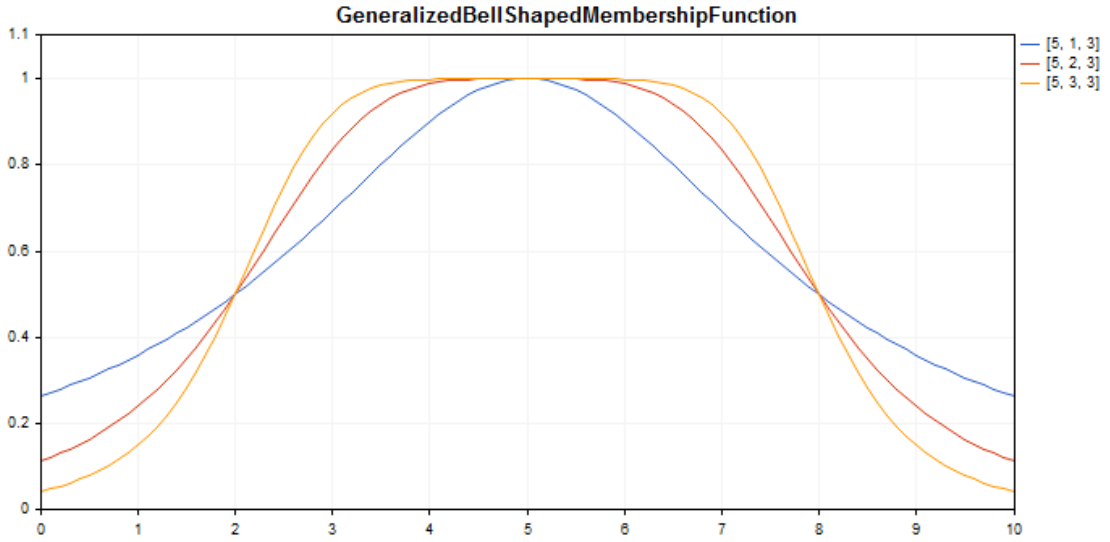
© 2000-2025, MetaQuotes Ltd.

CGeneralizedBellShapedMembershipFunction

A, B 및 C 매개변수로 일반화된 종 모양의 멤버십 함수를 구현하기 위한 클래스.

설명

일반화된 종 모양의 멤버십 함수 모양은 가우스 함수와 유사합니다. 함수는 매끄럽고 전체 정의 영역을 따라 0이 아닌 값을 취합니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CGeneralizedBellShapedMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

COBject
IMembershipFunction
CGeneralizedBellShapedMembershipFunction

클래스 메서드

클래스 메서드	설명
A	멤버스 함수 집중 비율을 가져오고 설정하기.
B	멤버십 함수 기울기 비율을 가져오고 설정하기.
C	멤버십 함수 최대 좌표를 가져오고 설정하기.

© 2000-2025, MetaQuotes Ltd.

クラス メソド	説明
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                      GeneralizedBellShapedMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd.     |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CGeneralizedBellShapedMembershipFunction func1(5, 1, 3);
CGeneralizedBellShapedMembershipFunction func2(5, 2, 3);
CGeneralizedBellShapedMembershipFunction func3(5, 3, 3);
//--- 멤버십 함수에 대한 래퍼 생성
double GeneralizedBellShapedMembershipFunction1(double x) { return(func1.GetValue(x));
double GeneralizedBellShapedMembershipFunction2(double x) { return(func2.GetValue(x));
double GeneralizedBellShapedMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"GeneralizedBellShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"GeneralizedBellShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("GeneralizedBellShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,
   graphic.CurveAdd(GeneralizedBellShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get method)

멤버십 함수 집중 비율을 가져오기.

```
double  A()
```

값 반환

집중 비율 값.

A (Set method)

멤버십 함수 농도 비율을 설정.

```
void  A(
   const double  a      // 집중 비율 값
   )
```

매개변수

a

[in] 멤버십 함수 집중 비율.

B (Get method)

멤버십 함수 기울기 비율 가져오기.

```
double  B()
```

값 반환

기울기 비율 값.

B (Set method)

멤버십 함수 기울기 비율 설정하기.

```
void  B(
   const double  b      // 기울기 비율 값
   )
```

© 2000-2025, MetaQuotes Ltd.

매개변수

b

[in] 멤버십 함수 기울기 비율.

C (Get method)

멤버십 함수 최대 좌표 가져오기.

```
double  C()
```

값 반환

멤버십 함수 최대 좌표.

C (Set method)

멤버십 함수 최대 좌표 설정하기.

```
void  C(
   const double  c      // 최대 좌표 값
   )
```

매개변수

c

[in] 멤버십 함수 최대 좌표.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

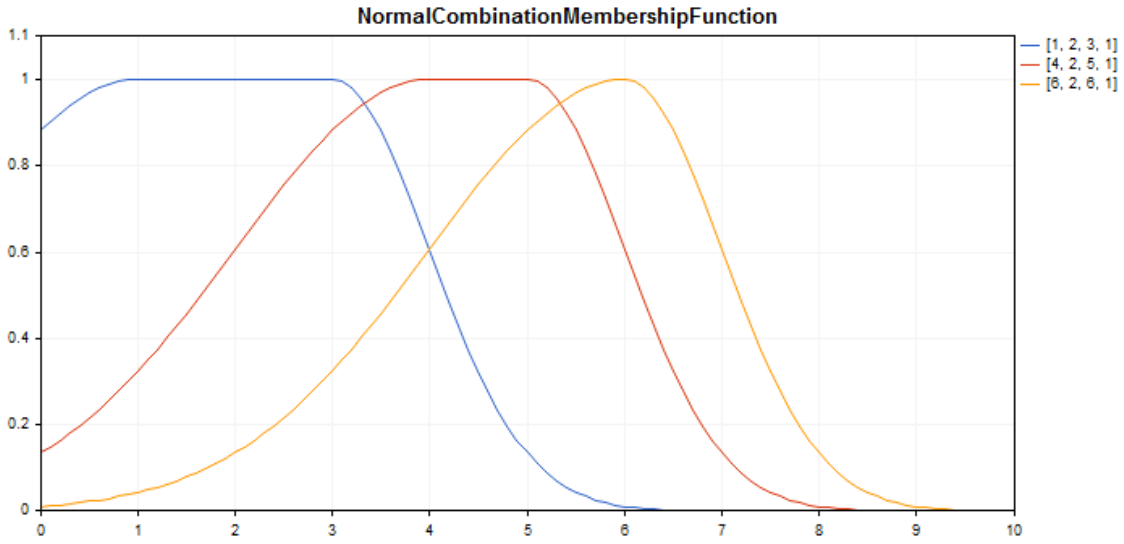
© 2000-2025, MetaQuotes Ltd.

CNormalCombinationMembershipFunction

B1, B2, Sigma1 및 Sigma2 매개변수로 양면 가우스 멤버십 함수를 구현하기 위한 클래스.

설명

양면 가우스 멤버십 함수는 가우스 분포를 사용하여 형성됩니다. 비대칭 멤버십 기능을 설정할 수 있습니다. 함수는 매끄럽고 전체 정의 영역을 따라 0이 아닌 값을 취합니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CNormalCombinationMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
IMembershipFunction
CNormalCombinationMembershipFunction

클래스 메서드

클래스 메서드	설명
B1	첫 번째 멤버십 함수 센터의 값을 가져오고 설정하기.
B2	두 번째 멤버 자격 기능 센터의 값을 가져오고 설정하기.

© 2000-2025, MetaQuotes Ltd.

클래스 메서드	설명
Sigma1	멤버십 함수 곡률의 첫 번째 매개변수를 가져오고 설정하기.
Sigma2	멤버십 함수 곡률의 두 번째 매개변수를 가져오고 설정하기.
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                          NormalCombinationMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CNormalCombinationMembershipFunction func1(1,2,3,1);
CNormalCombinationMembershipFunction func2(4,2,5,1);
CNormalCombinationMembershipFunction func3(6,2,6,1);
//--- 멤버십 함수에 대한 래퍼 생성
double NormalCombinationMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalCombinationMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalCombinationMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"NormalCombinationMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalCombinationMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalCombinationMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(NormalCombinationMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[1,
   graphic.CurveAdd(NormalCombinationMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[4,
   graphic.CurveAdd(NormalCombinationMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[6,
//--- X 축 속성 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B1 (Get method)

첫 번째 멤버십 함수 센터 값을 가져오기.

```
double  B1()
```

값 반환

첫 번째 멤버십 함수 센터 값.

B1 (Set method)

첫 번째 멤버십 함수 센터 값을 설정하기.

```
void  B1(
   const double  b1      // 첫 번째 센터 값
   )
```

매개변수

b

[in] 첫 번째 멤버십 함수 센터 값.

B2 (Get method)

두 번째 멤버십 함수 센터 값을 가져오기.

```
double  B2()
```

값 반환

두 번째 멤버십 함수 센터 값.

B2 (Set method)

© 2000-2025, MetaQuotes Ltd.

두 번째 멤버십 함수 센터 값을 설정하기.

```
void  B2(
   const double  b2      // 두 번째 센터 값
   )
```

매개변수

b2

[in] 두 번째 멤버십 함수 센터 값.

Sigma1 (Get method)

멤버십 함수 곡률의 첫 번째 매개변수 가져오기.

```
double  Sigma1()
```

값 반환

멤버십 함수 곡률의 첫 번째 매개변수 값.

Sigma1 (Set method)

멤버십 함수 곡률의 첫 번째 매개변수 값을 설정하기.

```
void  Sigma1(
   const double  sigma1      // 첫 번째 곡률 매개변수 값
   )
```

매개변수

sigma1

[in] 멤버십 함수 곡률의 첫 번째 매개변수.

Sigma2 (Get method)

멤버십 함수 곡률의 두 번째 매개변수 가져오기.

```
double  Sigma2()
```

값 반환

멤버십 함수 곡률의 두 번째 매개변수 값.

Sigma2 (Set method)

멤버십 함수 곡률의 두 번째 매개변수 설정하기.

```
void  Sigma2(
```

© 2000-2025, MetaQuotes Ltd.

```
   const double  sigma2      // 두 번째 곡률 매개변수 값
   )
```

매개변수

sigma2

[in] 멤버십 함수의 두 번째 곡률 매개변수.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

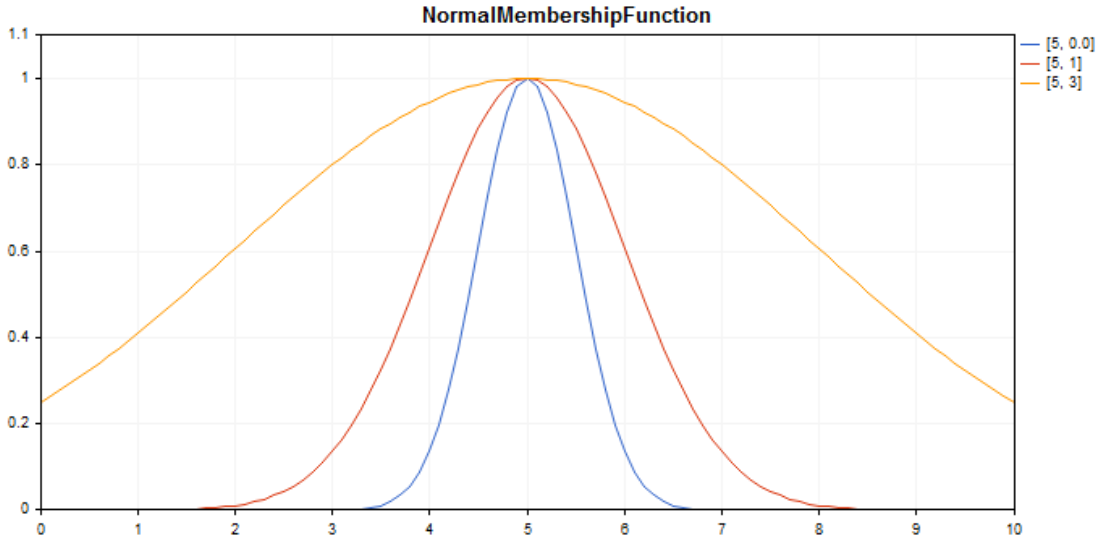
© 2000-2025, MetaQuotes Ltd.

CNormalMembershipFunction

B 및 Sigma 매개변수를 사용하여 대칭 가우스 멤버십 함수를 구현하기 위한 클래스.

설명

대칭 가우스 멤버십 함수는 가우스 분포를 사용하여 형성됩니다. 대칭 가우스 멤버십 함수는 가우스 분포를 사용하여 형성됩니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CNormalMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
IMembershipFunction
CNormalMembershipFunction

클래스 메서드

클래스 메서드	설명
B	멤버십 함수 센터를 가져오고 설정하기.
Sigma	멤버십 함수 곡률의 매개변수를 가져오고 설정하기.
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                                      NormalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CNormalMembershipFunction func1(5,0.5);
CNormalMembershipFunction func2(5,1);
CNormalMembershipFunction func3(5,3);
//--- 멤버십 함수에 대한 래퍼 생성
double NormalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"NormalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"NormalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("NormalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(NormalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[5, 0.0]");
   graphic.CurveAdd(NormalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[5, 1]");
   graphic.CurveAdd(NormalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[5, 3]");
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B (Get method)

멤버십 함수 센터를 가져오기.

```
double  B()
```

값 반환

멤버십 함수 센터 값.

B (Set method)

멤버십 함수 센터 값을 설정가기.

```
void  B(
   const double  b      // 함수 센터 값
   )
```

매개변수

b

[in] 멤버십 함수 센터 값.

Sigma (Get method)

멤버십 함수 곡률 매개변수를 가져오기.

```
double  Sigma()
```

값 반환

멤버십 함수 곡률 매개변수의 값.

Sigma (Set method)

멤버십 함수 곡률 매개변수 값을 설정하기.

```
void  Sigma(
   const double  sigma      // 곡률 매개변수 값
   )
```

매개변수

sigma

[in] 멤버십 함수 곡률 매개변수.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

© 2000-2025, MetaQuotes Ltd.

```
double  GetValue(
   const double  x      // 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

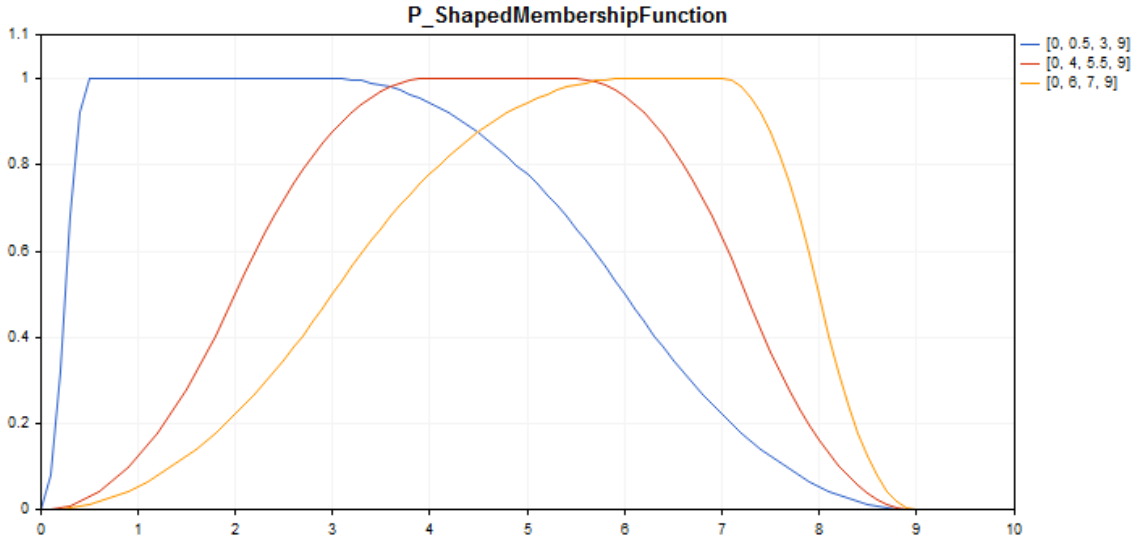
© 2000-2025, MetaQuotes Ltd.

CP_ShapedMembershipFunction

A,B,C 및 D 매개변수로 파이형 멤버십 함수를 구현하기 위한 클래스.

설명

파이형 멤버십 함수는 곡선 사다리꼴 형태를 가지고 있습니다. 이 함수는 비관적 퍼지 수 평가에서 낙관적 퍼지 수 평가로 평활 전환으로 비대칭 멤버십 함수를 설정하기 위해 사용됩니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CP_ShapedMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
  IMembershipFunction
    CP_ShapedMembershipFunction

클래스 메서드

클래스 메서드	설명
A	퍼지 집합 시작의 매개변수를 가져오고 설정하기.
B	퍼지 집합 코어의 첫 번째 매개변수를 가져오고 설정하기.

© 2000-2025, MetaQuotes Ltd.

클래스 메서드	설명
C	퍼지 집합 코어의 두 번째 매개변수를 가져오고 설정하기.
D	퍼지 집합 끝의 매개변수를 가져오고 설정하기.
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                                   P_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CP_ShapedMembershipFunction func1(0,0.5,3,9);
CP_ShapedMembershipFunction func2(0,4,5.5,9);
CP_ShapedMembershipFunction func3(0,6,7,9);
//--- 멤버십 함수에 대한 래퍼 생성
double P_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double P_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double P_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"P_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"P_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("P_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(P_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 0.5, 3,
   graphic.CurveAdd(P_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 4, 5.5,
   graphic.CurveAdd(P_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[0, 6, 7, 9
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get method)

퍼지 집합 시작의 매개변수 가져오기.

```
double  A()
```

값 반환

퍼지 집합 시작의 매개변수 .

A (Set method)

퍼지 집합 시작의 매개변수를 설정하기.

```
void  A(
   const double  a      // 퍼지 집합 시작의 매개변수
   )
```

매개변수

a

[in] 퍼지 집합 시작의 매개변수.

B (Get method)

퍼지 집합 코어의 첫 번째 매개변수를 가져오기.

```
double  B()
```

값 반환

퍼지 집합 코어의 첫 번째 매개변수.

B (Set method)

퍼지 집합 코어의 첫 번째 매개변수를 설정하기.

```
void  B(
   const double  b      // 퍼지 집합 코어 첫 번째 매개변수 값
   )
```

© 2000-2025, MetaQuotes Ltd.

매개변수

b

[in] 퍼지 집합 코어의 첫 번째 매개변수.

C (Get method)

퍼지 집합 코어의 두 번째 매개변수를 가져오기.

```
double  C()
```

값 반환

퍼지 집합 코어의 두 번째 매개변수.

C (Set method)

퍼지 집합 코어의 두 번째 매개변수를 설정하기.

```
void  C(
   const double  c      // 퍼지 집합 코어 두 번째 매개변수의 값
   )
```

매개변수

c

[in] 퍼지 집합 코어의 두 번째 매개변수.

D (Get method)

퍼지 집합 끝의 매개변수를 가져오기.

```
double  D()
```

값 반환

퍼지 집합 끝 매개변수의 값.

D (Set method)

퍼지 집합 끝의 매개변수를 설정하기.

```
void  D(
   const double  d      // 퍼지 집합 끝 매개변수의 값
   )
```

매개변수

d

[in] 퍼지 집합 끝 매개변수의 값.

GetValue

© 2000-2025, MetaQuotes Ltd.

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const double  x
   )
```

매개변수

x

[in] 멤버십 함수 인수

값 반환

멤버십 함수 값

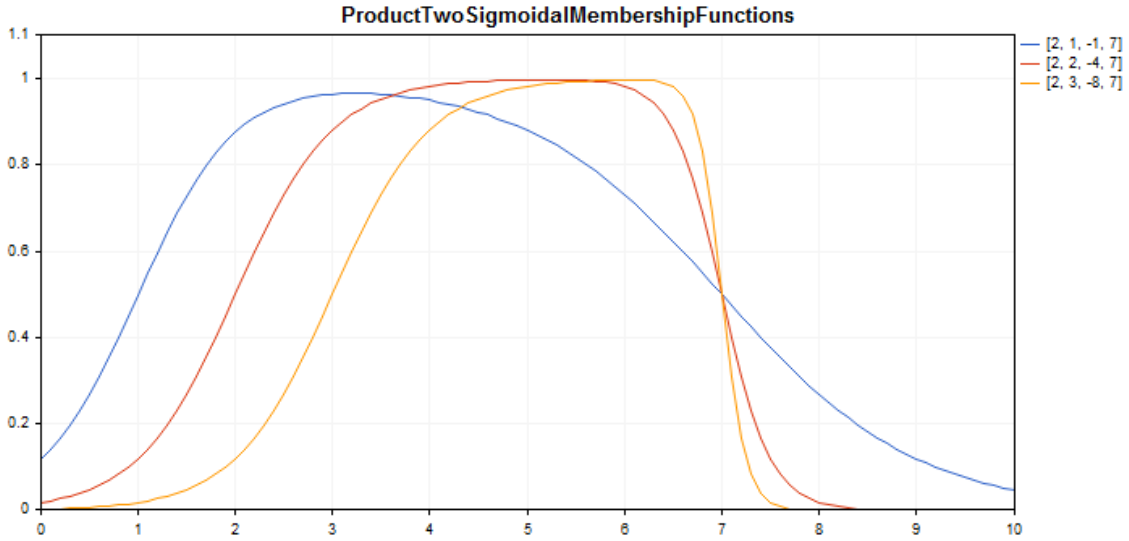
© 2000-2025, MetaQuotes Ltd.

CProductTwoSigmoidalMembershipFunction

멤버십 함수를 A1, A2, C1 및 C2 매개변수를 갖는 두 시그모이드 함수의 곱 형태로 구현하기 위한 클래스.

설명

평활 비대칭 함수를 설정하기 위해 두 시그모이드 멤버십 함수의 곱을 적용합니다. 인수 값으로 시작하는 값이 1과 같은 멤버십 함수를 만들 수 있습니다. 이러한 기능은 "short" 또는 "long"과 같은 용어를 설정해야 할 경우 적합합니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CProductTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject

IMembershipFunction

CProductTwoSigmoidalMembershipFunctions

클래스 메서드

클래스 메서드	설명
A1	첫 번째 멤버십 함수 기울기 비율을 가져오고 설정합니다.
A2	두 번째 멤버십 함수 기울기 비율을 가져와 설정하기.

© 2000-2025, MetaQuotes Ltd.

클래스 메서드	설명
C1	첫 번째 멤버십 함수 변곡 좌표 매개변수 가져오기.
C2	두 번째 멤버십 함수 변곡 좌표 매개변수 가져오기.
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                      ProductTwoSigmoidalMembershipFunctions.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CProductTwoSigmoidalMembershipFunctions func2(2,2,-4,7);
CProductTwoSigmoidalMembershipFunctions func3(2,3,-8,7);
//--- 멤버십 함수에 대한 래퍼 생성
double ProductTwoSigmoidalMembershipFunctions1(double x) { return(func1.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions2(double x) { return(func2.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"ProductTwoSigmoidalMembershipFunctions",0,30,30,780,380))
     {
      graphic.Attach(0,"ProductTwoSigmoidalMembershipFunctions");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("ProductTwoSigmoidalMembershipFunctions");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions1,0.0,10.0,0.1,CURVE_LINES,'
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions2,0.0,10.0,0.1,CURVE_LINES,'
   graphic.CurveAdd(ProductTwoSigmoidalMembershipFunctions3,0.0,10.0,0.1,CURVE_LINES,'
```

© 2000-2025, MetaQuotes Ltd.

```
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (Get method)

첫 번째 멤버십 함수 기울기 비율을 가져오기.

```
double  A1()
```

값 반환

첫 번째 멤버십 함수 기울기 비율.

A1 (Set method)

첫 번째 멤버십 함수 기울기 비율 설정하기.

```
void  A1(
   const double  a1      // 첫 번째 멤버십 함수 기울기 비율
   )
```

매개변수

a1

[in] 첫 번째 멤버십 함수 기울기 비율.

A2 (Get method)

두 번째 멤버십 함수 기울기 비율 가져오기.

```
double  A2()
```

값 반환

두 번째 멤버십 함수 기울기를 비율.

A2 (Set method)

두 번째 멤버십 함수 기울기 비율 설정하기.

```
void  A2(
```

© 2000-2025, MetaQuotes Ltd.

```
   const double  a2      // 두 번째 멤버십 함수 기울기 비율
   )
```

매개변수

a2

[in] 두 번째 멤버십 함수 기울기 비율.

C1 (Get method)

첫 번째 멤버십 함수 변곡 좌표 매개변수 가져오기.

```
double  C1()
```

값 반환

첫 번째 멤버십 함수 변곡 좌표.

C1 (Set method)

첫 번째 멤버십 함수 변곡 좌표를 설정하기.

```
void  C1(
   const double  c1      // 첫 번째 멤버십 함수 변곡 좌표
   )
```

매개변수

c1

[in] 첫 번째 멤버십 함수 변곡 좌표.

C2 (Get method)

두 번째 멤버십 함수 변곡 좌표 매개변수 가져오기.

```
double  C2()
```

값 반환

두 번째 멤버십 함수 변곡 좌표.

C2 (Set method)

두 번째 멤버십 합수 변곡 좌표를 설정하기.

```
void  C2(
   const double  c2      // 두 번째 멤버십 함수의 변곡 좌표
   )
```

매개변수

c2

[in] 두 번째 멤버십 함수 변곡 좌표.

© 2000-2025, MetaQuotes Ltd.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

© 2000-2025, MetaQuotes Ltd.

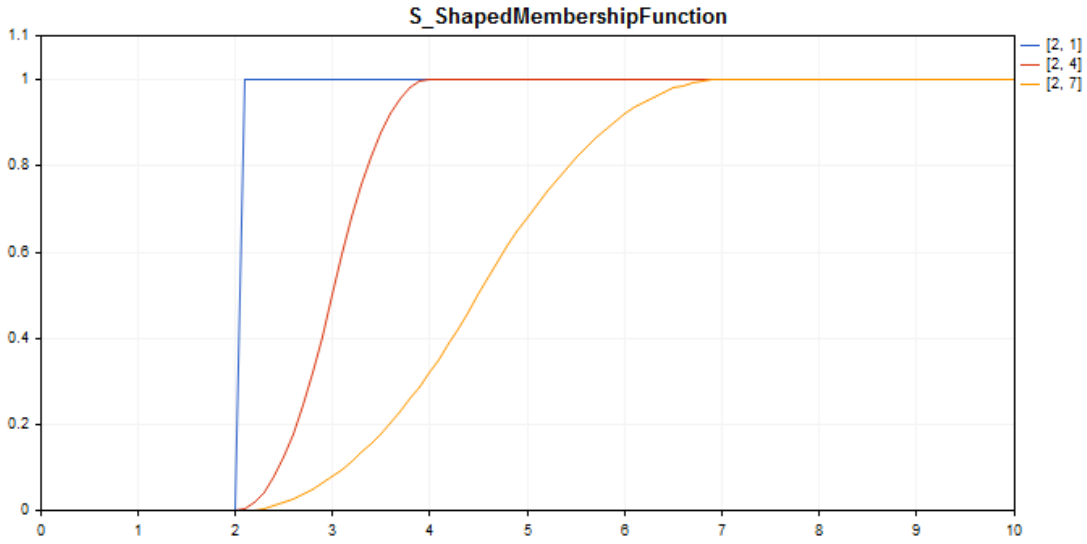
CS_ShapedMembershipFunction

A 및 B 매개변수를 사용하여 S와 같은 멤버십 함수를 구현하기 위한 클래스.

설명

함수는 S와 같은 두 매개변수 멤버십 함수를 설정합니다. 이는 0에서 1까지의 값을 갖는 비감소 함수입니다. A 및 B 매개변수는 함수가 0에서 1까지 비선형적 궤적으로 증가하는 간격을 정의합니다.

함수는 "매우 높음" 유형의 퍼지 집합을 나타냅니다(즉, 포화 상태의 감소하지 않는 멤버십 함수가 설정됨).



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CS_ShapedMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
  IMembershipFunction
    CS_ShapedMembershipFunction

클래스 메서드

클래스 메서드	설명
A	오름차 간격 시작의 매개변수를 가져오고 설정하기.

© 2000-2025, MetaQuotes Ltd.

クラス メンバー	説明
B	第一コアセットのパラメータを取得・設定する。
GetValue	指定された引数に基づいてメンバーシップ関数の値を計算する。

クラス CObject から継承されたメソッド

Prev, Prev, Next, Next, Save, Load, Type, Compare

例

```
//+------------------------------------------------------------------+
//|                                  S_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- メンバーシップ関数の作成
CS_ShapedMembershipFunction func1(2,1);
CS_ShapedMembershipFunction func2(2,4);
CS_ShapedMembershipFunction func3(2,7);
//--- メンバーシップ関数のラッパーの作成
double S_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double S_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double S_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| スクリプトプログラムの開始関数                                   |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- グラフィックの作成
  CGraphic graphic;
   if(!graphic.Create(0,"S_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"S_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("S_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 曲線の作成
   graphic.CurveAdd(S_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(S_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
   graphic.CurveAdd(S_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 7]");
//--- X 軸プロパティの設定
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get method)

오름차 간격 시작의 매개변수를 가져오기.

```
double  A()
```

값 반환

오름차 간격 시작 매개변수.

A (Set method)

오름차 간격 시작의 매개변수 설정하기.

```
void  A(
   const double  a      // 오름차 간격 시작 매개변수
   )
```

매개변수

a

[in] 오름차 간격 시작 매개 변수.

B (Get method)

퍼지 집합 코어의 첫 번째 매개변수를 가져오기.

```
double  B()
```

값 반환

퍼지 집합 코어의 첫 번째 매개변수.

B (Set method)

퍼지 집합 코어의 첫 번째 매개변수를 설정하기.

```
void  B(
   const double  b      // 퍼지 집합 코어의 첫 번째 매개 변수
   )
```

© 2000-2025, MetaQuotes Ltd.

매개변수

b

[in] 퍼지 집합 코어의 첫 번째 매개변수.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

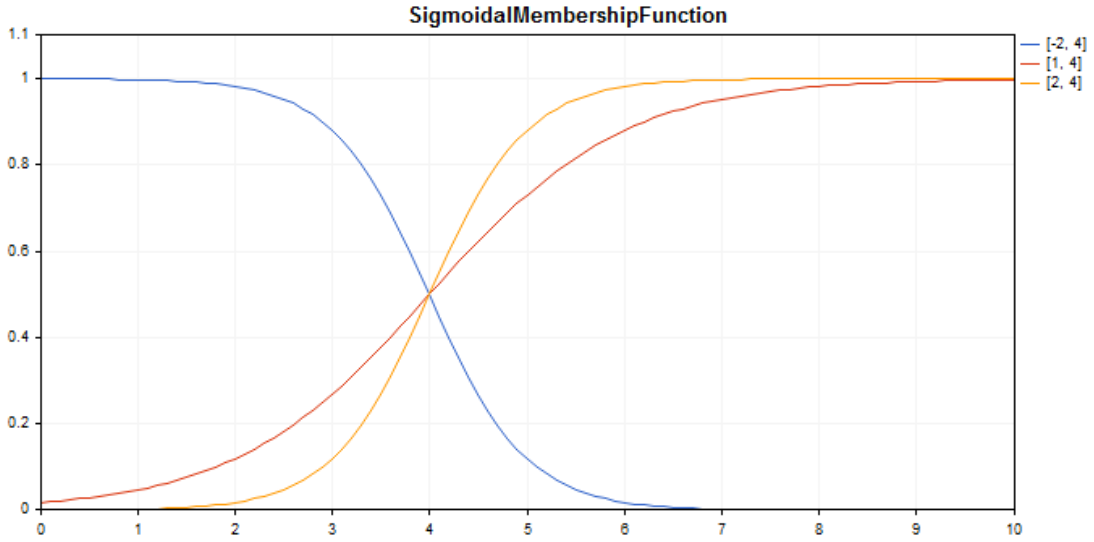
© 2000-2025, MetaQuotes Ltd.

CSigmoidalMembershipFunction

A 및 C 매개변수를 사용하여 시그모이드 멤버십 함수를 구현하기 위한 클래스.

설명

시그모이드 함수는 단조로운 멤버십 함수를 설정할 때 적용됩니다. 인수 값으로 시작하는 값이 1인 멤버십 함수를 생성할 수 있습니다. 이러한 기능은 "short" 또는 "long"과 같은 용어를 설정해야 할 경우 적합합니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CSigmoidalMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
  IMembershipFunction
    CSigmoidalMembershipFunction

클래스 메서드

클래스 메서드	설명
A	멤버십 함수 기울기 비율을 가져오고 설정하기.
C	멤버십 함수 변곡 좌표를 가져오고 설정하기.

© 2000-2025, MetaQuotes Ltd.

클래스 메서드	설명
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                                  SigmoidalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CSigmoidalMembershipFunction func1(-2, 4);
CSigmoidalMembershipFunction func2(1, 4);
CSigmoidalMembershipFunction func3(2, 4);
//--- 멤버십 함수에 대한 래퍼 생성
double SigmoidalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double SigmoidalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double SigmoidalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"SigmoidalMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"SigmoidalMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("SigmoidalMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(SigmoidalMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[-2, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[1, 4]");
   graphic.CurveAdd(SigmoidalMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 4]");
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get method)

멤버십 함수 기울기 비율 가져오기.

```
double  A()
```

값 반환

멤버십 함수 기울기 비율.

A (Set method)

멤버십 함수 기울기 비율 설정하기.

```
void  A(
   const double  a      // 첫 번째 멤버십 함수 기울기 비율
   )
```

매개변수

a

[in] 멤버십 함수 기울기 비율.

C (Get method)

멤버십 함수 변곡 좌표 매개변수 가져오기.

```
double  C()
```

값 반환

멤버십 함수 변곡 좌표.

C (Set method)

멤버십 함수 변곡 좌표 설정하기.

```
void  C(
   const double  c      // 멤버십 함수 변곡 좌표
   )
```

매개변수

c

© 2000-2025, MetaQuotes Ltd.

[in] 멤버십 함수 변곡 좌표.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

© 2000-2025, MetaQuotes Ltd.

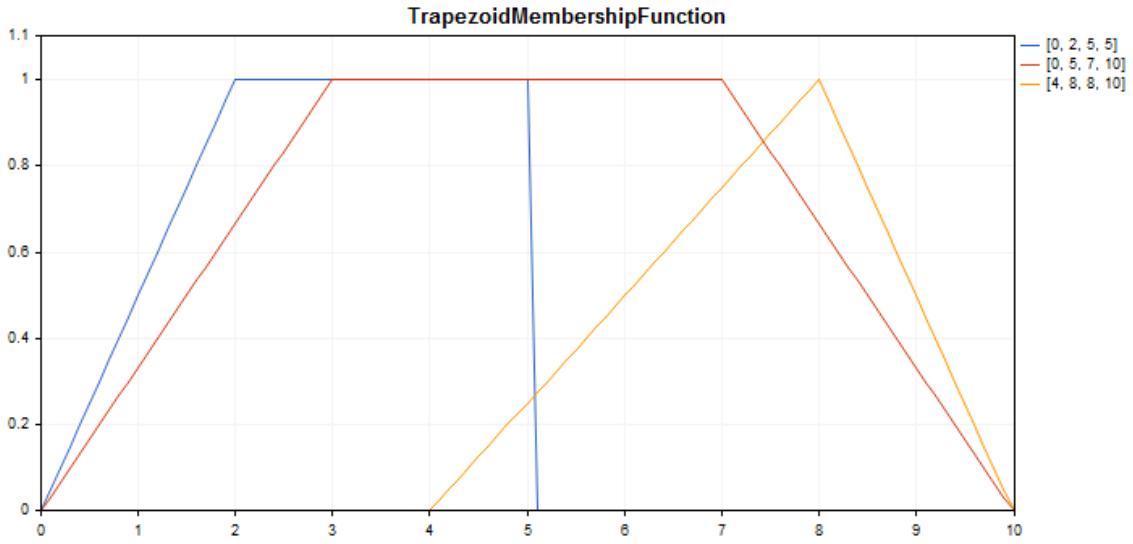
CTrapezoidMembershipFunction

X1, X2, X3 및 X4 매개변수를 사용하여 사다리꼴 멤버십 함수를 구현하기 위한 클래스.

설명

함수는 부분 선형 근사를 사용하여 형성됩니다. This is a generalization of the triangular function allowing you to assign a fuzzy set core as an interval. 이러한 멤버십 기능을 통해 낙관적/비관적 평가를 편리하게 해석할 수 있습니다.

함수는 변수의 비대칭 멤버십 함수를 특정 간격 내에 정의된 가장 중요한 값으로 설정하기 위해 사용됩니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CTrapezoidMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

COBject
IMembershipFunction
CTrapezoidMembershipFunction

클래스 메서드

클래스 메서드	설명
X1	X 축에서 첫 번째 점의 값을 가져오고 설정하기.

© 2000-2025, MetaQuotes Ltd.

クラス メソッド	설명
X2	X 축에서 두 번째 점의 값을 가져오고 설정하기.
X3	X 축에서 세 번째 점의 값을 가져오고 설정하기.
X4	X 축에서 네 번째 점의 값을 가져오고 설정하기.
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                                  TrapezoidMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CTrapezoidMembershipFunction func1(0,2,5,5);
CTrapezoidMembershipFunction func2(0,3,7,10);
CTrapezoidMembershipFunction func3(4,8,8,10);
//--- 멤버십 함수에 대한 래퍼 생성
double TrapezoidMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TrapezoidMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TrapezoidMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"TrapezoidMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TrapezoidMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TrapezoidMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(TrapezoidMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5, 5
   graphic.CurveAdd(TrapezoidMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 7, 1
   graphic.CurveAdd(TrapezoidMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[4, 8, 8, 1
//--- X 축 속성 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (Get method)

X 축에서 첫 번째 점의 값을 가져오기.

```
double  X1()
```

값 반환

X 축 첫 번째 점의 값.

X1 (Set method)

X 축에서 첫 번째 점의 값 설정하기.

```
void  X1(
   const double  x1      // X 축 첫 번째 점의 값
   )
```

매개변수

x1

[in] X 축 첫 번째 점의 값.

X2 (Get method)

X축에서 두 번째 점의 값 가져오기.

```
double  X2()
```

값 반환

X 축 두 번째 점의 값.

X2 (Set method)

X 축에서 두 번째 점의 값 설정하기.

```
void  X2(
   const double  x2      // X 축 두 번째 점의 값
```

© 2000-2025, MetaQuotes Ltd.

```
   )
```

매개변수

x2

[in] X 축 두 번째 점의 값.

X3 (Get method)

X 축에서 세 번째 점의 값 가져오기.

```
double  X3()
```

값 반환

X 축 세 번째 점의 값.

X3 (Set method)

X 축에서 세 번째 점의 값 설정하기.

```
void  X3(
   const double  x3      // X 축 세 번째 점의 값
   )
```

매개변수

x3

[in] X 축 세 번째 점의 값.

X4 (Get method)

X 축에서 네 번째 점의 값 가져오기.

```
double  X4()
```

값 반환

X 축의 네 번째 점의 값.

X4 (Set method)

X 축에서 네 번째 점의 값 설정하기.

```
void  X4(
   const double  x4      // X 축 네 번째 점의 값
   )
```

매개변수

x4

[in] X 축 네 번째 점의 값.

© 2000-2025, MetaQuotes Ltd.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

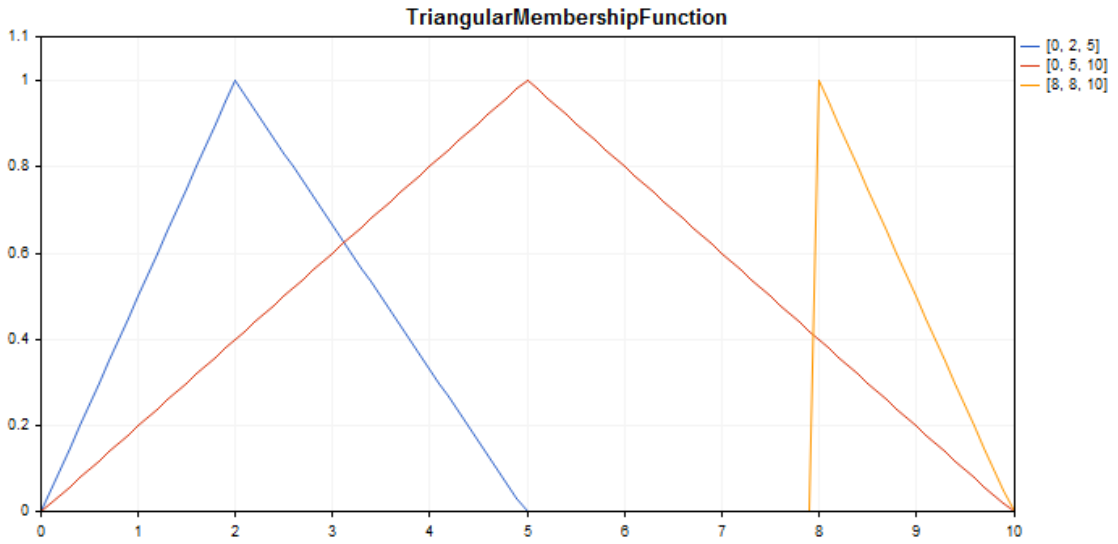
© 2000-2025, MetaQuotes Ltd.

CTriangularMembershipFunction

X1, X2 및 X3 매개변수를 사용하여 삼각 멤버십 함수를 구현하기 위한 클래스.

설명

함수는 삼각형의 형태로 멤버십 함수를 설정합니다. 이는 단순하고 가장 자주 적용되는 멤버십 함수입니다.



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CTriangularMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
  IMembershipFunction
    CTriangularMembershipFunction

클래스 메서드

클래스 메서드	설명
X1	X 축에서 첫 번째 점의 값을 가져오기.
X2	X축에서 두 번째 점의 값 가져오기.
X3	X 축에서 세 번째 점의 값 가져오기.
ToNormalMF	삼각 멤버십 함수를 가우스 함수로 변환하기.

© 2000-2025, MetaQuotes Ltd.

クラス メソッド	설명
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                                TriangularMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CTriangularMembershipFunction func1(0,2,5);
CTriangularMembershipFunction func2(0,5,10);
CTriangularMembershipFunction func3(8,8,10);
//--- 멤버십 함수에 대한 래퍼 생성
double TriangularMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TriangularMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TriangularMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"TriangularMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"TriangularMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("TriangularMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(TriangularMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[0, 2, 5]"
   graphic.CurveAdd(TriangularMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[0, 5, 10]
   graphic.CurveAdd(TriangularMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[8, 8, 10]
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (Get method)

X 축에서 첫 번째 점의 값을 가져오기.

```
double  X1()
```

값 반환

X 축 첫 번째 점의 값.

X1 (Set method)

X 축에서 첫 번째 점의 값 설정하기.

```
void  X1(
   const double  x1      // X 축 첫 번째 점의 값
   )
```

매개변수

x1

[in] X 축 첫 번째 점의 값.

X2 (Get method)

X축에서 두 번째 점의 값 가져오기.

```
double  X2()
```

값 반환

X 축 두 번째 점의 값.

X2 (Set method)

X 축에서 두 번째 점의 값 설정하기.

```
void  X2(
   const double  x2      // X 축 두 번째 점의 값
   )
```

© 2000-2025, MetaQuotes Ltd.

매개변수

x2

[in] X 축 두 번째 점의 값.

X3 (Get method)

X 축에서 세 번째 점의 값 가져오기.

```
double  X3()
```

값 반환

X 축 세 번째 점의 값.

X3 (Set method)

X 축에서 세 번째 점의 값 설정하기.

```
void  X3(
   const double  x3      // X 축 세 번째 점의 값
   )
```

매개변수

x3

[in] X 축 세 번째 점의 값.

ToNormalMF

삼각 멤버십 함수를 가우스 함수로 변환하기.

```
CNormalMembershipFunction*  ToNormalMF()
```

값 반환

가우스 멤버십 함수에 대한 포인터.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

© 2000-2025, MetaQuotes Ltd.

값 반환

멤버십 함수 값.

© 2000-2025, MetaQuotes Ltd.

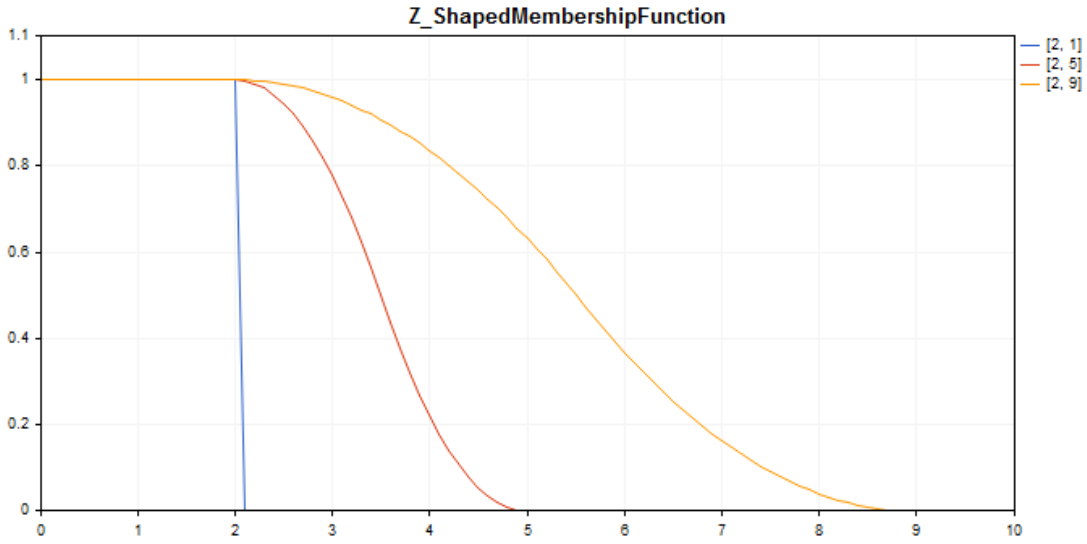
CZ_ShapedMembershipFunction

A 및 B 매개변수를 사용하여 z와 같은 멤버십 함수를 구현하기 위한 클래스.

설명

함수는 z와 같은 두 매개변수 멤버십 함수를 설정합니다. 1부터 0까지의 값을 갖는 비증가 멤버십 함수입니다. 함수 매개변수는 함수가 1에서 0까지 비선형 궤적으로 감소하는 간격을 정의합니다.

함수는 "매우 낮음" 유형의 퍼지 집합을 나타냅니다. 즉, 멤버십 함수가 증가하지 않도록 포화도로 설정합니다 .



샘플 코드(샘플 코드를 플롯팅하기 위한)가 아래에 표시됩니다.

선언

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject
IMembershipFunction
CZ_ShapedMembershipFunction

클래스 메서드

클래스 메서드	설명
A	내림차 간격 시작의 매개변수를 가져오고 설정하기.
B	내림차 간격 끝의 매개변수를 가져오고 설정하기.

© 2000-2025, MetaQuotes Ltd.

클래스 메서드	설명
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

예

```
//+------------------------------------------------------------------+
//|                                  Z_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- 멤버십 함수 생성
CZ_ShapedMembershipFunction func1(2,1);
CZ_ShapedMembershipFunction func2(2,5);
CZ_ShapedMembershipFunction func3(2,9);
//--- 멤버십 함수에 대한 래퍼 생성
double Z_ShapedMembershipFunction1(double x) { return(func1.GetValue(x)); }
double Z_ShapedMembershipFunction2(double x) { return(func2.GetValue(x)); }
double Z_ShapedMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 기능                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 그래픽 생성
  CGraphic 그래픽;
   if(!graphic.Create(0,"Z_ShapedMembershipFunction",0,30,30,780,380))
     {
      graphic.Attach(0,"Z_ShapedMembershipFunction");
     }
   graphic.HistoryNameWidth(70);
   graphic.BackgroundMain("Z_ShapedMembershipFunction");
   graphic.BackgroundMainSize(16);
//--- 곡선 생성
   graphic.CurveAdd(Z_ShapedMembershipFunction1,0.0,10.0,0.1,CURVE_LINES,"[2, 1]");
   graphic.CurveAdd(Z_ShapedMembershipFunction2,0.0,10.0,0.1,CURVE_LINES,"[2, 5]");
   graphic.CurveAdd(Z_ShapedMembershipFunction3,0.0,10.0,0.1,CURVE_LINES,"[2, 9]");
//--- X 축 속성 설정
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y 축 속성 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get method)

내림차 간격 시작의 매개변수를 가져오기.

```
double  A()
```

값 반환

내림차 간격 시작 매개변수.

A (Set method)

내림차 간격 시작의 매개변수 설정하기.

```
void  A(
   const double  a      // 내림차 간격 시작 매개변수
   )
```

매개변수

a

[in] 내림차 간격 시작 매개변수.

B (Get method)

내림차 간격 끝의 매개변수 가져오기.

```
double  B()
```

값 반환

내림차 간격 끝 매개변수.

B (Set method)

내림차 간격 끝의 매개변수 설정하기.

```
void  B(
   const double  b      // 내림차 간격 끝 매개변수
   )
```

매개변수

b

© 2000-2025, MetaQuotes Ltd.

[in] 내림차 간격 끝 매개변수.

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

© 2000-2025, MetaQuotes Ltd.

IMembershipFunction

모든 멤버십 함수 클래스의 기본 클래스.

선언

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

제목

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

상속 계층

CObject

IMembershipFunction

직계 자손

CCompositeMembershipFunction, CConstantMembershipFunction, CDifferencTwoSigmoidalMembershipFunction, CGeneralizedBellShapedMembershipFunction, CNormalCombinationMembershipFunction, CNormalMembershipFunction, CP_ShapedMembershipFunction, CProductTwoSigmoidalMembershipFunctions, CS_ShapedMembershipFunction, CSigmoidalMembershipFunction, CTrapezoidMembershipFunction, CTriangularMembershipFunction, CZ_ShapedMembershipFunction

클래스 메서드

클래스 메서드	설명
GetValue	지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

GetValue

지정한 인수를 기준으로 멤버 자격 함수의 값을 계산합니다.

```
double  GetValue(
   const  x      // 멤버십 함수 인수
   )
```

매개변수

x

[in] 멤버십 함수 인수.

값 반환

멤버십 함수 값.

© 2000-2025, MetaQuotes Ltd.

퍼지 시스템 규칙

퍼지 시스템(퍼지 논리 추론 시스템)은 **퍼지 규칙** 집합 및 퍼지 연산을 사용하여 입력의 현재 값에 해당하는 퍼지 집합의 형태로 결론을 수신하는 것입니다.

퍼지 규칙은 검사된 개체의 입력과 출력 사이의 관계를 결정합니다. 이 시스템의 규칙은 무제한입니다. 퍼지 규칙의 일반화된 형식은 다음과 같다:

if *rule condition*, **then** *rule conclusion*.

*규칙 조건*은 개체의 현재 상태를 설명합니다. *규칙 결론*은 규칙 조건이 어떤 영향을 미치는지 설명합니다.

퍼지 시스템에 대한 규칙 클래스	설명
CMamdaniFuzzyRule	Mamdani 알고리즘에 대한 퍼지 논리 규칙을 구현하기 위한 클래스
CSugenoFuzzyRule	스게노 알고리즘에 대한 퍼지 논리 규칙을 구현하기 위한 클래스
CSingleCondition	클래스는 "퍼지 변수 — 퍼지 항" 쌍으로 표현되는 퍼지 조건을 설정합니다.
CConditions	클래스는 연산자에 의해 서로 연결된 퍼지 조건 집합을 정의합니다.
CGenericFuzzyRule	두 가지 유형의 퍼지 규칙을 구현하기 위한 기본 클래스.

© 2000-2025, MetaQuotes Ltd.

CMamdaniFuzzyRule

Mamdani형 퍼지 추론 — 퍼지 시스템의 두 가지 기본 유형 중 하나. 퍼지 항을 사용하여 출력 변수 값을 설정.

설명

Mamdani 알고리즘에 대한 퍼지 논리 규칙은 다음과 같이 설명될 수 있다:

$$if(X_1\ is\ a_1) \wedge (X_2\ is\ a_2) \wedge \ldots \wedge (X_n\ is\ a_n)\ then\ (Y\ is\ d)(W)$$

위치:

- X = (X1, X2, X3 ... Xn) — 입력 변수의 벡터;
- Y — 출력 변수;
- a = (a1, a2, a3 ... an) — 입력 변수 값의 벡터;
- d — 출력 변수 값;
- W — 규칙 가중치.

선언

```
class CMamdaniFuzzyRule : public CGenericFuzzyRule
```

제목

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

상속 계층

CObject
  IParsableRule
    CGenericFuzzyRule
      CMamdaniFuzzyRule

클래스 메서드

클래스 메서드	설명
Conclusion	Mamdani 퍼지 규칙 결론을 가져오고 설정하기
Weight	Mamdani 퍼지 규칙 가중치를 가져오고 설정하기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CGenericFuzzyRule 클래스에서 상속된 메서드

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion (Get method)

© 2000-2025, MetaQuotes Ltd.

Mamdani 퍼지 규칙 결론 가져오기

```
CSingleConditon*  Conclusion()
```

값 반환

Mamdani 퍼지 규칙 결론.

Conclusion (Set method)

Mamdani 퍼지 규칙 결론 설정하기.

```
void  Conclusion(
   CSingleConditon*  value      // Mamdani 퍼지 규칙 결론
   )
```

매개변수

값

[in] Mamdani 퍼지 규칙 결론.

Weight (Get method)

Mamdani 퍼지 규칙 가중치 가져오기.

```
double  Weight()
```

값 반환

Mamdani 퍼지 규칙 가중치.

Weight (Set method)

Mamdani 퍼지 규칙 가중치를 설정하기.

```
void  Weight(
   const double  value      // Mamdani 퍼지 규칙의 가중치
   )
```

매개변수

값

[in] Mamdani 퍼지 규칙의 가중치.

© 2000-2025, MetaQuotes Ltd.

CSugenoFuzzyRule

스게노(Sugeno) 유형의 퍼지 추론 — 퍼지 시스템의 두 가지 기본 유형 중 하나입니다. 출력 변수 값은 입력 변수의 선형 조합으로 설정됩니다.

설명

Mamdani 규칙과 달리, 입력 변수 값은 퍼지 항에 의해서가 아니라 입력 매개변수로부터의 선형 삼수에 의해 설정됩니다. 스게노 알고리즘의 퍼지 논리 규칙은 다음과 같이 설명될 수 있습니다:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

위치:

- X = (X1, X2, X3 ... Xn) — 입력 변수의 벡터;
- Y — 출력 변수;
- a = (a1, a2, a3 ... an) — 입력 변수 값의 벡터;
- b = (b1, b2, b3 ... bn) — 출력 값에 대한 선형 함수의 자유 항 비율
- W — 규칙 가중치.

선언

```
class CSugenoFuzzyRule : public CGenericFuzzyRule
```

제목

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

상속 계층

CObject
  IParsableRule
    CGenericFuzzyRule
      CSugenoFuzzyRule

클래스 메서드

클래스 메서드	설명
Conclusion	스게노 퍼지 규칙 결론을 가져오고 설정하기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CGenericFuzzyRule 클래스에서 상속된 메서드

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion (Get method)

© 2000-2025, MetaQuotes Ltd.

스게노 퍼지 규칙 결론 가져오기.

```
CSingleConditon*  Conclusion()
```

값 반환

스게노 퍼지 규칙 결ㄹ론.

Conclusion (Set method)

스게노 퍼지 규칙 결론 설정하기.

```
void  Conclusion(
   CSingleCondition*  value      // 스게노 퍼지 규칙의 결론
   )
```

매개변수

값

[in] 스게노 퍼지 규칙의 결론.

© 2000-2025, MetaQuotes Ltd.

CSingleCondition

클래스는 "퍼지 변수 —퍼지 항" 쌍으로 표현되는 퍼지 조건을 설정합니다.

설명

퍼지 조건에 따르면, 하나의 변수는 하나의 항에 해당합니다. 퍼지 조건은 다음과 같은 식으로 설명할 수 있습니다: *X는*

위치:

- X는 퍼지 변수입니다;
- a 는 퍼지 변수 값(퍼지 항)입니다.

선언

```
class CSingleCondition : public ICondition
```

제목

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

상속 계층

CObject
  ICondition
    CSingleCondition

직계 자손

CFuzzyCondition

클래스 메서드

클래스 메서드	설명
Not	이 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그를 가져오고 설정하기.
Term	이 조건에 대한 퍼지 항을 가져오고 설정하기.
Var	이 조건에 대한 퍼지 변수를 가져오고 설정하기.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

Not (Get method)

이 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그를 가져오기.

```
bool Not()
```

값 반환

© 2000-2025, MetaQuotes Ltd.

플래그 값.

Not (Set method)

이 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그를 설정하기.

```
void  Not(
   bool  not      // 플래그 값
   )
```

매개변수

부정

[in] 플래그 값.

Term (Get method)

주어진 조건에 대한 퍼지 항 가져오기.

```
INamedValue* Term()
```

값 반환

주어진 조건에 대한 퍼지 항.

Term (Set method)

주어진 조건에 대한 퍼지 항 설정하기.

```
void  Term(
   INamedValue*&  value      // 주어진 조건에 대한 퍼지 항
   )
```

매개변수

값

[in] 주어진 조건에 대한 퍼지 항.

Var (Get method)

주어진 조건에 대한 퍼지 변수 가져오기.

```
INamedVariable* Var()
```

값 반환

주어진 조건에 대한 퍼지 변수.

Var (Set method)

주어진 조건에 대한 퍼지 변수 설정하기

```
void  Var(
   INamedVariable*&  value      // 주어진 조건에 대한 퍼지 변수
```

© 2000-2025, MetaQuotes Ltd.

```
)
```

매개변수

값

[in] 퍼지 변수.

© 2000-2025, MetaQuotes Ltd.

CConditions

클래스는 연산자에 의해 서로 연결된 퍼지 조건 집합을 정의합니다.

설명

운영자에 의해 서로 연결된 퍼지 조건 집합은 다음과 같이 설명될 수 있습니다:

$$(X_1 \; is \; a_1) \wedge (X_2 \; is \; a_2) \wedge \ldots \wedge (X_n \; is \; a_n)$$

위치:

- X = (X1, X2, X3 ... Xn) —입력 변수의 벡터;
- a = (a1, a2, a3 ... an) —입력 변수 값의 벡터.

이 예에서 *and* 연산자가 사용됩니다. 게다가 *or* 연산자는 이 클래스에서 사용할 수 있습니다.

선언

```
class CConditions : public ICondition
```

제목

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

상속 계층

CObject
  ICondition
    CConditions

클래스 메서드

클래스 메서드	설명
ConditionsList	모든 조건 목록을 가져오기
Not	이러한 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그를 가져오고 설명합니다
Op	조건 번들 연산자의 유형을 가져오고 설정합니다.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

ConditionsList

모든 조건 목록 가져오기.

```
CList*  ConditionsList()
```

값 반환

© 2000-2025, MetaQuotes Ltd.

모든 조건 목록.

Not (Get method)

이러한 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그 가져오기.

```
bool Not()
```

값 반환

플래그 값.

Not (Set method)

이러한 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그 설정하기

```
void  Not(
   bool  not      // 플래그 값
   )
```

매개변수

부정

[in] 플래그 값.

Op (Get method)

조건 번들 연산자의 유형 가져오기. *and* 및 *or* 연산자를 사용할 수 있습니다.

```
OperatorType Op()
```

값 반환

조건 번들 연산자 유형.

Op (Set method)

조건 번들 연산자 설정하기. *and* 및 *or* 연산자를 사용할 수 있습니다.

```
void  Op(
   OperatorType  op     // 조건 번들 연산자 유형
   )
```

매개변수

op

[in] 조건 번들 연산자 유형.

© 2000-2025, MetaQuotes Ltd.

CGenericFuzzyRule

두 유형의 퍼지 규칙에 대한 기본 클래스.

선언

```
class CGenericFuzzyRule : public IParsableRule
```

제목

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

상속 계층

CObject

IParsableRule

CGenericFuzzyRule

직계 자손

CMamdaniFuzzyRule, CSugenoFuzzyRule

클래스 메서드

클래스 메서드	설명
Conclusion	퍼지 규칙 결론을 가져오고 설명하기
Condition	퍼지 규칙에 대한 'if' 조건(조건 집합)을 가져오고 설정하기
CreateCondition	지정된 매개변수별로 퍼지 규칙에 대한 조건을 생성하기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

Conclusion (Get method)

퍼지 규칙의 결론 가져오기.

```
CSingleConditon*  Conclusion()
```

값 반환

퍼지 규칙의 결론.

Conclusion (Set method)

퍼지 규칙 결론 설정하기.

```
virtual void  Conclusion(
   CSingleConditon*  value      // 퍼지 규칙의 결론
   )
```

© 2000-2025, MetaQuotes Ltd.

매개변수

값

[in] 퍼지 규칙의 결론.

Condition (Get method)

퍼지 규칙에 대한 'if' 조건(조건 집합)을 가져오기.

```
CConditons*  Condition()
```

값 반환

퍼지 조건(조건 집합).

Condition (Set method)

퍼지 규칙에 대한 'if' 조건(조건 집합)을 설정하기.

```
void  Condition(
   CConditons*  value      // 퍼지 규칙에 대한 'if' 조건(조건 집합)
   )
```

매개변수

값

[in] 퍼지 조건(조건 집합).

CreateCondition

지정된 매개변수별로 퍼지 규칙에 대한 조건을 생성하기.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,      // 퍼지 변수
   CFuzzyTerm*      term,     // 퍼지 항
   )
```

매개변수

var

[in] 퍼지 변수.

항

[in] 퍼지 항.

값 반환

퍼지 규칙 상태.

CreateCondition

지정된 매개변수별로 퍼지 규칙에 대한 조건을 생성하기.

© 2000-2025, MetaQuotes Ltd.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,      // 퍼지 변수
   CFuzzyTerm*      term,     // 퍼지 항
   bool             not,      // 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그
   )
```

매개변수

var

[in] 퍼지 변수.

항

[in] 퍼지 항.

부정

[in] 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그.

값 반환

퍼지 규칙 상태.

CreateCondition

지정된 매개변수별로 퍼지 규칙에 대한 조건을 생성하기.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable* var,      // 퍼지 변수
   CFuzzyTerm*     term,     // 퍼지 항
   bool            not,      // 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그
   HedgeType       hedge     // 조건 번들 유형
   )
```

매개변수

var

[in] 퍼지 변수.

항

[in] 퍼지 항.

부정

[in] 조건에 부정을 적용해야 하는지 여부를 나타내는 플래그.

헤지

[in] Condition bundle type.

값 반환

퍼지 규칙 상태.

© 2000-2025, MetaQuotes Ltd.

퍼지 시스템 변수

퍼지(언어) 변수는 **퍼지 시스템에 적용됩니다**. 이것들은 자연어 또는 인공어에서의 단어나 단어 조합의 값들입니다.

언어 변수는 퍼지 집합으로 구성됩니다. 퍼지 집합을 정의할 때 각 직업에 대해 퍼지 변수의 특성과 수가 변경됩니다.

Class	설명
CFuzzyVariable	일반 퍼지 변수를 만들기 위한 클래스.
CSugenoVariable	퍼지 스게노 유형 변수를 생성하기 위한 클래스.

© 2000-2025, MetaQuotes Ltd.

CFuzzyVariable

일반 퍼지 변수를 만들기 위한 클래스.

설명

여기서 퍼지 변수는 다음 매개변수를 사용하여 생성됩니다:

- 최대 변수 값;
- 최소 변수 값;
- 퍼지 변수 이름;
- 항 집합 (언어 변수가 수신할 수 있는 모든 가능한 값의 집합).

선언

```
class CFuzzyVariable : public CNamedVariableImpl
```

제목

```
#include <Math\Fuzzy\fuzzyvariable.mqh>
```

상속 계층

CObject
INamedValue
INamedVariable
CNamedVariableImpl
CFuzzyVariable

클래스 메서드

클래스 메서드	설명
AddTerm	퍼지 변수에 퍼지 항 하나를 추가하기.
GetTermByName	지정된 이름으로 퍼지 항을 가져오기.
Max	퍼지 변수 최대 값을 가져오고 설정하기.
Min	퍼지 변수에 대한 최소 값을 가져오고 설정하기.
Terms	지정된 퍼지 변수에 대한 퍼지 항의 목록을 가져오고 설정하기.
Values	지정된 퍼지 변수에 대한 퍼지 항의 목록을 가져오고 설정하기.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CNamedVariableImpl 클래스에서 상속된 메서드

Name, Name

© 2000-2025, MetaQuotes Ltd.

AddTerm

퍼지 변수에 퍼지 항 하나를 추가하기.

```
void  AddTerm(
   CFuzzyTerm*&  term      // 퍼지 항
   )
```

매개변수

항

[in] 퍼지 항.

GetTermByName

지정된 이름으로 퍼지 항을 가져오기.

```
CFuzzyTerm*  GetTermByName(
   const string  name      // 퍼지 항 이름
   )
```

매개변수

이름

[in] 퍼지 항 이름.

값 반환

이름이 지정된 퍼지 항.

Max (Get method)

퍼지 변수 최대 값을 가져오기.

```
double  Max()
```

값 반환

퍼지 변수의 최대 변수.

Max (Set method)

퍼지 변수의 최대 값을 설정하기.

```
void  Max(
   const double  max      // 퍼지 변수의 최대 값
   )
```

매개변수

max

[in] 버지 변수의 최대 값.

© 2000-2025, MetaQuotes Ltd.

Min (Get method)

퍼지 변수의 최소 값 가져오기.

```
double  Min()
```

값 반환

퍼지 변수의 최소 값.

Max (Set method)

퍼지 변수의 최소 값 설정하기.

```
void  Min(
   const double  min      // 퍼지 변수의 최소 값
   )
```

매개변수

min

[in] 퍼지 변수의 최소 값.

Terms (Get method)

주어진 퍼지 변수에 대한 퍼지 항 목록 가져오기.

```
CList*  Terms()
```

값 반환

주어진 퍼지 변수의 퍼지 항 목록.

Terms (Set method)

주어진 퍼지 변수의 퍼지 항 목록 설정하기.

```
void  Terms(
   CList*&  terms      // 주어진 변수의 퍼지 항 목록
   )
```

매개변수

terms

[in] 주어진 퍼지 변수의 퍼지 항 목록.

Values

주어진 퍼지 변수에 대한 퍼지 항 목록 가져오기.

```
CList*  Values()
```

값 반환

© 2000-2025, MetaQuotes Ltd.

주어진 변수의 퍼지 항 목록.

© 2000-2025, MetaQuotes Ltd.

CSugenoVariable

퍼지 스게노 유형 변수를 생성하기 위한 클래스.

설명

퍼지 스게노 유형 번수는 용어가 아닌 선형 함수의 집합에 의해 설정되기 때문에 일반적인 언어 변수와 다릅니다.

선언

```
class CSugenoVariable : public CNamedVariableImpl
```

제목

```
#include <Math\Fuzzy\sugenovariable.mqh>
```

상속 계층

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CSugenoVariable

클래스 메서드

클래스 메서드	설명
Functions	퍼지 스게노 변수의 선형 함수 목록 가져오기.
GetFuncByName	지정한 이름으로 선형 함수 가져오기.
Values	퍼지 스게노 변수의 선형 함수 목록 가져오기.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CNamedVariableImpl 클래스에서 상속된 메서드

Name, Name

Functions

퍼지 스게노 변수의 선형 함수 목록 가져오기.

```
CList*  Functions()
```

값 반환

선형 함수 목록.

© 2000-2025, MetaQuotes Ltd.

GetFuncByName

지정한 이름으로 선형 함수 가져오기.

```
ISugenoFunction*  GetFuncByName(
   const string  name      // 선형 함수 이름
   )
```

매개변수

이름

[in] 선형 함수 이름.

값 반환

지정된 이름의 선형 함수.

Values

퍼지 스게노 변수의 선형 함수 목록 가져오기.

```
CList*  Values()
```

값 반환

퍼지 스게노 변수의 선형 함수 목록.

© 2000-2025, MetaQuotes Ltd.

CFuzzyTerm (fuzzy terms)

퍼지 항을 구현하기 위한 클래스.

설명

항은 항 집합의 임의 요소입니다. 항은 다음 두 가지 요소로 정의됩니다:

- 퍼지 항 이름;
- 멤버십 함수.

선언

```
class CFuzzyTerm : public CNamedValueImpl
```

제목

```
#include <Math\Fuzzy\fuzzyterm.mqh>
```

상속 계층

CObject
  INamedValue
    CNamedValueImpl
      CFuzzyTerm

클래스 메서드

클래스 메서드	설명
MembershipFunction	퍼지 항의 멤버십 함수를 가져오기.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CNamedValueImpl 클래스에서 상속된 메서드

Name, Name

© 2000-2025, MetaQuotes Ltd.

MembershipFunction

퍼지 항의 멤버십 함수를 가져오기.

```
IMembershipFunction*  MembershipFunction()
```

값 반환

멤버십 함수

© 2000-2025, MetaQuotes Ltd.

Fuzzy systems

퍼지 시스템(또는 *퍼지 모델*)은 퍼지 논리에 기반한 계산의 수학적 모델입니다. 이러한 모델의 구성은 연구 대상이 형식화가 약학 정확한 수학적 설명이 너무 복잡하거나 알려지지 않은 경우에 적용할 수 있습니다.

모델 제작의 진행은 크게 세 단계로 나눌 수 있습니다:

1. 모델의 입력 및 출력 특성에 대한 정의.
2. 기술 기반 구축.
3. 퍼지 추론 방법 중 하나 선택 (Mamdani 및 스게노).

첫 번째 단계는 결과 2에 직접 영향을 미치고 모델의 향후 작동을 결정합니다.

기술 기반(*규칙 기반*)은 검사된 개체의 입력과 출력 사이의 관계를 정의하는 "if, then" 유형의 퍼지 규칙 집합입니다.

규칙 조건은 개체의 현재 상태와 **규칙 결론**(이 조건이 개체에 미치는 영향)을 설명합니다.

각 규칙에는 두 가지 유형의 항과 결론이 있을 수 있습니다:

1. 단순 (Csinglcond 링크) — 하나의 퍼지 변수를 포함;
2. 복합 (Cconditions 링크) — 여러 퍼지 변수를 포함.

시스템의 각 규칙에는 **가중치**라는 모델에 있는 규칙의 중요성이 있습니다. 가중치 인자는 [0,1] 범위 내의 규칙에 할당됩니다.

생성된 지식 기반에 따라 모델에 대한 퍼지 추론 시스템이 결정됩니다. **퍼지 논리 추론**은 지식 기반 및 퍼지 연산을 사용하여 입력의 현재 값에 해당하는 퍼지 집합의 형태로 결론을 수신하는 것입니다. 퍼지 추론의 두 가지 주요 유형은 Mamdani 및 스게노입니다.

© 2000-2025, MetaQuotes Ltd.

Mamdani 시스템

Mamdani 시스템의 출력 변수 값은 퍼지 항을 사용하여 설정됩니다.

설명

Mamdani 알고리즘에 대한 퍼지 논리 규칙은 다음과 같이 설명될 수 있다:

$$if(X_1 \ is \ a_1) \wedge (X_2 \ is \ a_2) \wedge \ldots \wedge (X_n \ is \ a_n) \ then \ (Y \ is \ d)(W)$$

위치:

- X = (X1, X2, X3 ... Xn) — 입력 변수의 벡터;
- Y — 출력 변수;
- a = (a1, a2, a3 ... an) — 입력 변수 값의 벡터;
- d — 출력 변수 값;
- W — 규칙 가중치.

클래스 메서드

클래스 메서드	설명
AggregationMethod	조건 집계 유형을 설정하기
Calculate	시스템에 대한 퍼지 추론을 계산하기
DefuzzificationMethod	디퍼지화 메서드 유형을 설정하기
EmptyRule	현지 시스템을 기준으로 빈 퍼지 Mamdani 규칙을 생성하기
ImplicationMethod	시스템 함축 연산자의 유형을 설정하기
Output	퍼지 Mamdani 출력 변수 목록을 가져오기.
OutputByName	지정된 이름의 퍼지 Mamdani 출력 변수 가져오기.
ParseRule	지정된 선을 기준으로 퍼지 Mamdani 규칙을 생성하기.
Rules	퍼지 Mamdani 규칙 목록을 반환하기.

CGenericFuzzySystem 클래스에서 상속된 메서드

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

AggregationMethod

조건 집계 메서드 유형을 설정하기.

```
void  AggregationMethod(
   AggregationMethod  value      // 집계 메서드 유형
   )
```

매개변수

© 2000-2025, MetaQuotes Ltd.

값

[in] 조건 집계 메서드 유형.

Calculate

시스템에 대한 퍼지 추론을 계산하기

```
CList*  Calculate(
   CList*  inputValues      // 입력 데이터
   )
```

매개변수

inputValues

[in] 계산을 위한 입력 데이터.

값 반환

계산 결과.

DefuzzificationMethod

디퍼지화 메서드 유형을 설정하기.

```
void  DefuzzificationMethod(
   DefuzzificationMethod  value      // 디퍼지화 메서드 유형.
   )
```

매개변수

값

[in] 디퍼지화 메서드 유형.

EmptyRule

현지 시스템을 기준으로 빈 퍼지 Mamdani 규칙을 생성하기

```
CMamdaniFuzzyRule*  EmptyRule()
```

값 반환

퍼지 Mamdani 규칙.

ImplicationMethod

조건 함축 연산자의 유형을 설정하기.

```
void  ImplicationMethod(
   ImplicationMethod  value      // 함축 연산자 유형
   )
```

매개변수

© 2000-2025, MetaQuotes Ltd.

값

[in] 조건 함축 연산자의 유형.

Output

퍼지 Mamdani 출력 변수 목록을 가져오기.

```
CList*  Output()
```

값 반환

퍼지 변수의 목록.

OutputByName

지정된 이름의 퍼지 Mamdani 출력 변수 가져오기.

```
CFuzzyVariable*  OutputByName(
   const string  name      // 퍼지 변수 이름
   )
```

매개변수

이름

[in] 퍼지 변수 이름.

값 반환

지정된 이름의 퍼지 Mamdani 변수.

ParseRule

지정된 선을 기준으로 퍼지 Mamdani 규칙을 생성하기.

```
CMamdaniFuzzyRule*  ParseRule(
   const string  rule      // 퍼지 규칙의 문자열 표현
   )
```

매개변수

rule

[in] 퍼지 Mamdani 규칙의 문자열 표현.

값 반환

퍼지 Mamdani 규칙.

Rules

퍼지 Mamdani 규칙 목록을 반환하기.

```
CList*  Rules()
```

© 2000-2025, MetaQuotes Ltd.

값 반환

퍼지 Mamdani 규칙 목록.

© 2000-2025, MetaQuotes Ltd.

스게노(Sugeno) 시스템

스게노 퍼지 논리 시스템은 퍼지 시스템의 두 가지 기본 유형 중 하나입니다. 출력 변수 값은 입력 변수의 선형 조합으로 설정됩니다.

설명

Mamdani 규칙과 달리, 입력 변수 값은 퍼지 항에 의해서가 아니라 엔트리의 선형 함수에 의해 설정됩니다. 스게노 알고리즘의 퍼지 논리 규칙은 다음과 같이 설명될 수 있습니다:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

위치:

- X = (X1, X2, X3 ... Xn) — 입력 변수의 벡터;
- Y — 출력 변수;
- a = (a1, a2, a3 ... an) — 입력 변수 값의 벡터;
- b = (b1, b2, b3 ... bn) — 출력 값에 대한 선형 함수의 자유 항 비율
- W — 규칙 가중치.

클래스 메서드

클래스 메서드	설명
Calculate	시스템에 대한 퍼지 추론을 계산하기
CreateSugenoFunction	시스템에 대한 선형 스게노 함수를 생성합니다.
EmptyRule	현재 시스템을 기반으로 빈 퍼지 스게노 규칙을 만들기.
Output	퍼지 스게노 출력 변수 목록 가져오기.
OutputByName	지정한 이름으로 퍼지 스게노 출력 변수 가져오기.
ParseRule	지정된 선을 기준으로 퍼지 스게노 규칙을 생성하기.
Rules	퍼지 규칙 목록 반환하기.

CGenericFuzzySystem 클래스에서 상속된 메서드

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

Calculate

시스템에 대한 퍼지 추론을 계산하기

```
CList*  Calculate(
   CList*&  inputValues      // 입력 데이터
   )
```

매개변수

inputValues

© 2000-2025, MetaQuotes Ltd.

[in] 계산을 위한 입력 데이터.

값 반환

계산 결과.

CreateSugenoFunction

시스템에 대한 선형 스게노 함수를 생성합니다.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // 함수 이름
   const double& coeffs[]       // 함수 비율
   )
```

매개변수

이름

[in] 함수 이름.

coeffs[]

[in] 함수 비율.

값 반환

선형 스게노 함수.

참고

비율 배열의 크기는 입력 수와 같거나 해당 수를 1씩 초과할 수 있습니다. 첫 번째 경우 스게노 선형 함수의 자유 항은 0이고 두 번째 경우 마지막 비율과 같습니다.

CreateSugenoFunction

시스템에 대한 선형 스게노 함수를 생성합니다.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,          // 함수 이름
   CList*&       coeffs,        // 퍼지 변수 페어 목록 - 그 비율
   const double  constValue     // 함수 자유 항 비율
   )
```

매개변수

이름

[in] 함수 이름.

coeffs[]

[in] 함수 비율.

값 반환

선형 스게노 함수.

© 2000-2025, MetaQuotes Ltd.

EmptyRule

현재 시스템을 기반으로 빈 퍼지 스게노 규칙을 만들기.

```
CSugenoFuzzyRule*  EmptyRule()
```

값 반환

퍼지 스게노 규칙.

Output

퍼지 스게노 출력 변수 목록 가져오기.

```
CList*  Output()
```

값 반환

퍼지 변수의 목록.

OutputByName

지정한 이름으로 퍼지 스게노 출력 변수 가져오기.

```
CSugenoVariable*  OutputByName(
   const string  name      // 퍼지 변수 이름
   )
```

매개변수

이름

퍼지 변수 이름.

값 반환

지정된 이름의 퍼지 스게노 함수.

ParseRule

지정된 선을 기준으로 퍼지 스게노 규칙을 생성하기.

```
CSugenoFuzzyRule*  ParseRule(
   const string  rule      // 퍼지 스게노 규칙의 문자열 표현
   )
```

매개변수

rule

[in] 퍼지 스게노 규칙의 문자열 표현.

값 반환

퍼지 스게노 규칙.

© 2000-2025, MetaQuotes Ltd.

Rules

퍼지 규칙 목록 반환하기.

```
CList*  Rules()
```

값 반환

퍼지 규칙 목록.

© 2000-2025, MetaQuotes Ltd.

OpenCL 프로그램 작업을 위한 클래스

COpenCL 클래스는 OpenCL 함수 작업을 용이하게 하기 위한 래퍼입니다. 어떤 경우에는 GPU를 사용하면 계산 속도를 상당히 높일 수 있습니다.

float 및 double 값을 기반으로 한 계산을 위한 클래스 사용의 예는 MQL5₩Scripts₩Examples₩OpenCL₩ 폴더의 해당 하위 디렉토리에서 확인할 수 있습니다. OpenCL 프로그램의 소스 코드는 MQL5₩Scripts₩Examples₩OpenCL₩Double₩Kernels 및 MQL5₩Scripts₩Examples₩OpenCL₩Float₩Kernels 하위 디렉토리에 있습니다.

- MatrixMult.mq5 – 전역 및 로컬 메모리를 사용한 행렬 곱셈의 예
- BitonicSort.mq5 – GPU에서 배열 요소의 병렬 정렬 예
- FFT.mq5 – 고속 푸리에(Fourier) 변환 계산의 예
- Wavelet.mq5 – Morlet 웨이블릿을 사용한 데이터의 웨이블릿 변환 예.

OpenCL의 소스코드를 별도의 CL 파일에 작성하는 것이 좋으며, 이 소스 코드는 나중에 리소스 변수를 사용하여 MQL5 프로그램에 포함될 수 있습니다.

선언

```
class COpenCL
```

제목

```
#include <OpenCL\OpenCL.mqh>
```

클래스 메서드

Name	설명
BufferCreate	지정된 인덱스에 OpenCL 버퍼를 생성하기
BufferFree	지정된 인덱스에서 버퍼 삭제
BufferFromArray	값 배열에서 지정된 인덱스에 버퍼를 생성하기
BufferRead	지정된 인덱스의 OpenCL 버퍼를 배열로 읽기
BufferWrite	지정된 인덱스의 버퍼에 값 배열 쓰기
Execute	지정된 인덱스로 OpenCL 커널 실행
GetContext	Returns handle of the OpenCL 컨텍스트의 핸들을 반환
GetKernel	지정한 인덱스에서 커널 개체의 핸들을 반환
GetKernelName	지정한 인덱스에 있는 커널 개체의 이름을 반환
GetProgram	OpenCL 프로그램의 핸들을 반환
Initialize	OpenCL 프로그램을 초기화
KernelCreate	지정된 인덱스에서 OpenCL 프로그램에 진입점을 생성
KernelFree	지정된 인덱스에서 OpenCL 시작 함수를 제거

© 2000-2025, MetaQuotes Ltd.

Name	설명
SetArgument	지정된 인덱스에서 OpenCL 함수의 매개 변수를 설정
SetArgumentBuffer	지정된 인덱스에서 OpenCL 함수의 매개변수를 설정
SetArgumentLocalMemory	지정된 인덱스에서 OpenCL 함수의 매개변수를 설정
SetBuffersCount	버퍼 수를 설정
SetKernelsCount	커널 개체의 수 설정
Shutdown	OpenCL 프로그램 언로드
SupportDouble	장치에서 부동 소수점 데이터 유형이 지원되는지 확인

© 2000-2025, MetaQuotes Ltd.

BufferCreate

지정된 인덱스에 OpenCL 버퍼 생성하기.

```
bool  BufferCreate(
   const int   buffer_index,      // 버퍼 인덱스
   const uint  size_in_bytes,     // 버퍼 사이즈(바이트)
   const uint  flags              // 버퍼 속성을 정의하는 플래그의 조합
   );
```

매개변수

buffer_index

[in] 인덱스 버퍼.

size_in_bytes

[in] 버퍼 크기(바이트).

flags

[in] 플래그의 조합을 사용하여 설정된 버퍼 속성.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

BufferFree

지정된 인덱스에서 버퍼를 삭제.

```
bool  BufferFree(
   const int  buffer_index      // 버퍼 인덱스
   );
```

매개변수

buffer_index

[in] 인덱스 버퍼.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

BufferFromArray

값 배열에서 지정한 인덱스에 버퍼를 생성하기.

```
template<typename T>
bool  BufferFromArray(
   const int   buffer_index,          // 버퍼 인덱스
   T           &data[],               // 값의 배열
   const uint  data_array_offset,     // 값 배열의 오프셋(바이트)
   const uint  data_array_count,      // 쓸 배열의 값 수
   const uint  flags                  // 버퍼 속성을 정의하는 플래그의 조합
   );
```

매개변수

buffer_index

[in] 인덱스 버퍼.

&data[]

[in] OpenCL 버퍼에 기록할 값의 배열.

data_array_offset

[in] 바이트 단위 값 배열에서 오프셋 되며, 여기서 값 쓰기가 시작됩니다.

data_array_count

[in] 기록할 값의 수.

flags

[in] 플래그의 조합을 사용하여 설정된 버퍼 속성.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

BufferRead

지정된 인덱스의 OpenCL 버퍼를 배열로 읽기.

```
template<typename T>
bool  BufferRead(
   const int   buffer_index,          // 버퍼 인덱스
   T           &data[],               // 값의 배열
   const uint  cl_buffer_offset,      // OpenCL 버퍼에서 오프셋(바이트)
   const uint  data_array_offset,     // 배열 요소의 이동
   const uint  data_array_count       // 버퍼에서 읽을 값의 수
   );
```

매개변수

buffer_index

[in] 인덱스 버퍼.

&data[]

[in] OpenCL 버퍼 값을 얻기 위한 배열.

cl_buffer_offset

[in] 값 읽기를 시작할 OpenCL 버퍼의 오프셋(바이트).

data_array_offset

[in] OpenCL 버퍼의 값을 쓸 배열의 첫 번째 요소의 인덱스.

data_array_count

[in] 읽을 값의 수.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

BufferWrite

지정된 인덱스의 버퍼에 값 배열을 쓰기.

```
template<typename T>
bool  BufferWrite(
   const int   buffer_index,          // 버퍼 인덱스
   T           &data[],               // 값의 배열
   const uint  cl_buffer_offset,      // OpenCL 버퍼에서 오프셋(바이트)
   const uint  data_array_offset,     // 배열 요소의 이동
   const uint  data_array_count       // 쓸 배열의 값 수
   );
```

매개변수

buffer_index

[in] 인덱스 버퍼.

&data[]

[in] OpenCL 버퍼에 기록할 값의 배열.

cl_buffer_offset

[in] 값 쓰기를 시작하는 OpenCL 버퍼의 오프셋(바이트).

data_array_offset

[in] OpenCL 버퍼에 배열 값이 기록되는 배열의 첫 번째 요소 인덱스₩

data_array_count

[in] 기록할 값의 수.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Execute

지정된 인덱스에서 OpenCL 프로그램을 실행.

```
bool  Execute(
   const int   kernel_index,           // 커널의 인덱스
   const int   work_dim,               // 작업 공간의 치수
   const uint  &work_offset[],         // 작업 공간의 초기 오프셋
   const uint  &work_size[]            // 작업의 총 수
   );
```

로컬 그룹의 지정된 인덱스 및 작업 수를 사용하여 OpenCL 커널을 실행.

```
bool  Execute(
   const int   kernel_index,           // 커널의 인덱스
   const int   work_dim,               // 작업 공간의 치수
   const uint  &work_offset[],         // 작업 공간의 초기 오프셋
   const uint  &work_size[],           // 작업의 총 수
   const uint  &local_work_size[]      // 로컬 그룹의 작업 수
   );
```

매개변수

kernel_index
  [in] 커널 개체의 인덱스.

work_dim
  [in] 작업 공간의 치수.

&work_offset[]
  [in][out] 작업 공간의 초기 오프셋. 참조로 통과.

&work_size[]
  [in][out] 작업 하위 세트의 크기. 참조로 통과.

&local_work_size[]
  [in][out] 그룹의 로컬 작업 하위 세트의 크기. 참조로 통과.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

GetContext

OpenCL 컨텍스트 핸들을 반환.

```
int  GetContext();
```

값 반환

OpenCL 컨텍스트 핸들.

© 2000-2025, MetaQuotes Ltd.

GetKernel

지정된 인덱스에서 커널 개체의 핸들 반환.

```
int  GetKernel(
   const int  kernel_index      // 커널 인덱스
   );
```

매개변수

kernel_index

[in] 커널 개체의 인덱스.

값 반환

커널 개체의 핸들.

© 2000-2025, MetaQuotes Ltd.

GetKernelName

지정된 인덱스에서 커널 개체 이름 반환.

```
string  GetKernelName(
   const int  kernel_index      // 커널 인덱스
   );
```

매개변수

kernel_index

[in] 커널 개체의 인덱스.

값 반환

커널 개체 이름.

© 2000-2025, MetaQuotes Ltd.

GetProgram

OpenCL 프로그램 핸들 반환.

```
int  GetProgram();
```

값 반환

OpenCL 프로그램 핸들.

© 2000-2025, MetaQuotes Ltd.

Initialize

OpenCL 프로그램 초기화.

```
bool  Initialize(
   const string  program,           // OpenCL 프로그램 핸들
   const bool    show_log=true      // 로그 보관
   );
```

매개변수

프로그램

[in] OpenCL 프로그램 핸들.

show_log=true

[in] 메시지 기록 사용.

값 반환

초기화가 성공하면, true 반환. 그렇지 않으면 false 반환.

© 2000-2025, MetaQuotes Ltd.

KernelCreate

지정된 인덱스에서 OpenCL 프로그램에 진입점을 생성.

```
bool  KernelCreate(
   const int     kernel_index,     // 커널 인덱스
   const string  kernel_name       // 커널 이름
   );
```

매개변수

kernel_index

[in] 커널 개체의 인덱스.

kernel_name

[in] 커널 개체 이름.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

KernelFree

지정한 인덱스에서 OpenCL 시작 함수를 제거.

```
bool  KernelFree(
   const int  kernel_index      // 커널 인덱스
   );
```

매개변수

kernel_index

[in] 커널 개체의 인덱스.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SetArgument

지정된 인덱스에서 OpenCL 함수의 매개변수를 설정.

```
template<typename T>
bool  SetArgument(
   const int  kernel_index,     // 커널의 인덱스
   const int  arg_index,        // 함수 인수의 인덱스
   T          value             // 소스 코드
   );
```

매개변수

kernel_index

[in] 커널 개체의 인덱스.

arg_index

[in] 함수 인수의 인덱스.

값

[in] 함수 인수의 값.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SetArgumentBuffer

OpenCL 버퍼를 지정된 인덱스에서 OpenCL 함수의 매개변수로 설정.

```
bool  SetArgumentBuffer(
   const int  kernel_index,     // 커널의 인덱스
   const int  arg_index,        // 함수 인수의 인덱스
   const int  buffer_index      // 버퍼 인덱스
   );
```

매개변수

kernel_index

[in] 커널 개체의 인덱스.

arg_index

[in] 함수 인수의 인덱스.

buffer_index

[in] 인덱스 버퍼.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SetArgumentLocalMemory

지정된 인덱스의 OpenCL 기능에 대한 로컬 메모리의 매개변수를 설정.

```
bool  SetArgumentLocalMemory(
   const int  kernel_index,          // 커널의 인덱스
   const int  arg_index,             // 함수 인수의 인덱스
   const int  local_memory_size      // 로컬 메모리의 크기
   );
```

매개변수

kernel_index

[in] 커널 개체의 인덱스.

arg_index

[in] 함수 인수의 인덱스.

local_memory_size

[in] 로컬 메모리의 크기.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SetBuffersCount

버퍼의 숫자 설정.

```
bool  SetBuffersCount(
   const int  total_buffers      // 버퍼의 숫자
   );
```

매개변수

total_buffers

[in] 버퍼의 총 수.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SetKernelsCount

커널 개체의 수 설정.

```
bool  SetKernelsCount(
   const int  total_kernels      // 커널의 수
   );
```

매개변수

total_kernels

[in] 커널의 총 수.

값 반환

성공적 실행의 경우 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Shutdown

OpenCL 프로그램 언로드.

```
void  Shutdown();
```

값 반환

반환 값 없음.

© 2000-2025, MetaQuotes Ltd.

SupportDouble

장치에서 부동 소수점 데이터 유형이 지원되는지 확인.

```
bool  SupportDouble();
```

값 반환

장치가 부동 소수점 데이터 형식을 지원하는 경우 true 반환.

© 2000-2025, MetaQuotes Ltd.

기본 클래스 CObject

클래스 CObject는 MQL5 표준 라이브러리를 구성하기 위한 기본 클래스입니다.

설명

클래스 CObject를 사용하면 모든 하위 항목이 링크된 목록의 일부가 될 수 있습니다. 또한 하위 클래스에서 추가 구현하기 위한 여러 가상 메서드가 식별됩니다.

선언

```
class CObject
```

제목

```
#include <Object.mqh>
```

상속 계층

CObject

직계 후손

CAccountInfo, CArray, CChart, CChartObject, CCurve, CDealInfo, CDictionary_Obj_Double, CDictionary_Obj_Obj, CDictionary_String_Obj, CExpertBase, CFile, CHistoryOrderInfo, CList, COrderInfo, CPositionInfo, CString, CSymbolInfo, CTerminalInfo, CTrade, CTreeNode, CWnd, ICondition, IExpression, IMembershipFunction, INamedValue, IParsableRule

그룹별 클래스 메서드

특성	
Prev	이전 항목의 값 가져오기
Prev	이전 항목의 값 설정하기
Next	후속 요소의 값을 가져오기
Next	다음 요소를 설정하기
메서드 배교하기	
virtual Compare	다른 개체와의 비교 결과를 반환하기
입력/출력	
virtual Save	파일에 개체 쓰기
virtual Load	파일에서 개체 읽기
virtual Type	개체 유형 반환하기

© 2000-2025, MetaQuotes Ltd.

Prev

목록에서 이전 요소에 대한 포인터 가져오기.

```
CObject*  Prev()
```

값 반환

목록의 이전 요소에 대한 포인터. 항목이 먼저 나열되면 NULL을 반환합니다.

예시:

```
//--- CObject::Prev() 예시
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("개체 생성 오류");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("개체 생성 오류");
      delete object_first;
      return;
     }
   //--- 상호 연결 설정
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- use prev object
   CObject *object=object_second.Prev();
   //--- 개체 삭제하기
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Prev

포인터를 목록의 이전 요소로 설정하기.

```
void  Prev(
   CObject*  object      // 목록의 이전 요소에 대한 포인터
  \)
```

매개변수

개체

[in] 목록의 이전 요소에 대한 포인터의 새 값.

예시:

```
//--- CObject::Prev(CObject*) 예시
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("개체 생성 오류");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("개체 생성 오류");
      delete object_first;
      return;
     }
   //--- 상호 연결 설정
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- 개체 사용
   //--- ...
   //--- 개체 삭제하기
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Next

목록의 다음 요소에 대한 포인터 가져오기.

```
CObject*  Next()
```

값 반환

목록이 다음요소에 대한 포인터. 목록의 마지막 요소인 경우 NULL을 반환합니다.

예시:

```
//--- CObject::Next() 예시
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("개체 생성 오류");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("개체 생성 오류");
      delete object_first;
      return;
     }
   //--- 상호 연결 설정
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- 다음 요소 사용하기
   CObject *object=object_first.Next();
   //--- 개체 삭제하기
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Next

포인터를 목록의 다음 요소로 설정하기.

```
void  Next(
   CObject*  object      // 포인터를 목록의 다음 요소로 설정
  \)
```

매개변수

개체

[in] 목록의 다음 요소에 대한 포인터의 새 값.

예시:

```
//--- example for CObject::Next(CObject*)
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("개체 생성 오류");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("개체 생성 오류");
      delete object_first;
      return;
     }
   //--- 상호 연결 설정
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- 개체 사용
   //--- ...
   //--- 개체 삭제하기
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Compare

목록 요소의 데이터와 다른 목록 요소의 데이터를 비교하기.

```
virtual int  Compare(
   const CObject*  node,      // 요소
   const int       mode=0     // 변형
   ) const
```

매개변수

노드

[in] 비교할 목록 요소에 대한 포인터

mode=0

[in] 비교 변형

값 반환

목록 요소가 동일한 경우 0, 목록 요소가 비교에 사용되는 요소(노드)보다 작으면 -1, 목록 요소가 비교에 사용되는 요소(노드)보다 크면 1.

참고

CObject 클래스의 Compare() 메서드는 항상 0을 반환하며 아무 작업도 수행하지 않습니다. 파생 클래스의 데이터를 비교하려면 Compare(...) 방법을 구현해야 합니다. 다변 비교를 구현할 때 '모드' 매개변수를 사용해야 합니다.

예시:

```
//--- CObject::Compare(...) 예시
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("개체 생성 오류");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("개체 생성 오류");
      delete object_first;
      return;
     }
   //--- 상호 연결 설정
   object_first.Next(object_second);
```

© 2000-2025, MetaQuotes Ltd.

```
   object_second.Prev(object_first);
   //--- 개체 비교하기
   int result=object_first.Compare(object_second);
   //--- 개체 삭제하기
   delete object_first;
   delete object_second;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

목록 요소 데이터를 파일에 저장하기.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
  \)
```

매개변수

file_handle

[in] FileOpen () 함수를 사용하여 이전에 연 binary 파일의 핸들

값 반환

성공적으로 완료하면 true, 오류이면 false.

참고

CObject 클래스의 Save(int) 메서드는 항상 'true'를 반환하며 어떠한 작업도 수행하지 않습니다. 파생 클래스의 데이터를 파일에 저장하려면 Save(int) 메서드를 구현해야 합니다.

예시:

```
//--- CObject::Save(int) 예시
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("개체 생성 오류");
      return;
     }
   //--- 개체 데이터 설정
   //--- . . .
   //--- 파일 열기
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("File save: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
```

© 2000-2025, MetaQuotes Ltd.

```
      FileClose(file_handle);
     }
   delete object;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 목록 요소 데이터를 로드하기.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
  \)
```

매개변수

file_handle

[in] FileOpen() 함수를 사용하여 이전에 연 binary 파일의 핸들.

값 반환

성공적으로 완료되면 true, 오류이면 false.

참고

CObject 클래스의 Load(int) 메서드는 항상 'true'를 반환하며 어떠한 작업도 수행하지 않습니다. 파일에서 파생 클래스의 데이터를 로드하려면 Load(int) 메서드를 구현해야 합니다.

예시:

```
//--- CObject::Load(int) 예시
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("개체 생성 오류");
      return;
     }
   //--- 파일 열기
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("File load: Error %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 개체 사용
   //--- . . .
   delete object;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

유형 식별자 가져오기.

```
virtual int  Type() const
```

값 반환

유형 식별자 (CObject에는 0).

예시:

```
//--- CObject::Type() 예시
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object=new CObject;
   //---
   object=new CObject;
   if(object ==NULL)
     {
      printf("개체 생성 오류");
      return;
     }
   //--- 개체 유형 가져오기
   int type=object.Type();
   //--- 개체 삭제하기
   delete object;
  }
```

© 2000-2025, MetaQuotes Ltd.

데이터 구조

이 섹션에는 다양한 데이터 구조(배열, 연결 목록 등) 작업에 대한 기술 세부 사항과 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 수록되어 있습니다.

데이터 구조의 클래스를 사용하면 복합 데이터 구조를 포함한 다양한 형식의 사용자 지정 데이터 저장소를 만들 때 시간이 절약됩니다.

MQL5 표준 라이브러리(데이터 세트의 경우)는 Include₩Arrays 폴더에 있는 터미널이 작업 디렉토리에 배치됩니다.

데이터 배열

동적 데이터 배열 클래스를 사용하면 다차원 배열을 포함한 다양한 형식의 사용자 지정 데이터 저장소를 만들 때 시간이 절약됩니다.

MQL5 표준 라이브러리(데이터 배열의 경우)은 Include₩Arrays 폴더에 있는 터미널의 작업 디렉토리에 있습니다.

클래스	설명
CArray	동적 데이터 배열의 기본 클래스
CArrayChar	char 또는 uchar 변수의 동적 배열
CArrayShort	short 또는 ushort 변수의 동적 배열
CArrayInt	int 또는 uint 변수의 동적 배열
CArrayLong	long 또는 ulong 변수의 동적 배열
CArrayFloat	float 변수의 동적 배열
CArrayDouble	double 변수의 동적 배열
CArrayString	string 변수의 동적 배열
CArrayObj	CObject 포인터의 동적 배열
CList	CObject 클래스의 인스턴스 목록 및 하위 항목 목록과 함께 작업할 수 있는 기능을 제공
CTreeNode	CTree binary 트리의 노드로 작업할 수 있는 기능을 제공
CTree	CTreeNode 클래스 인스턴스 및 하위 인스턴스의 binary 트리와 함께 작업할 수 있는 기능을 제공

© 2000-2025, MetaQuotes Ltd.

CArray

CArray 클래스는 동적 변수 배열의 기본 클래스입니다.

Description

클래스 CArray는 동적 변수 배열(메모리 할당, 정렬, 파일 작업)에서 동작합니다.

Declaration

```
class CArray : public CObject
```

Title

```
#include <Arrays\Array.mqh>
```

상속 계층

CObject

CArray

직계 종속

CArrayChar, CArrayDouble, CArrayFloat, CArrayInt, CArrayLong, CArrayObj, CArrayShort, CArrayString

그룹별 클래스 메서드

Attributes	
Step	배열의 증분 크기를 가져옵니다
Step	배열의 증분 크기를 설정합니다
Total	배열의 요소 수를 가져옵니다
Available	추가 메모리 할당 없이 사용할 수 있는 배열의 사용 가능한 요소 수를 가져옵니다
Max	메모리 재할당 없이 배열의 가능한 최대 크기를 가져옵니다
IsSorted	지정된 정렬 모드를 사용하여 정렬되는 배열의 플래그를 가져옵니다
SortMode	배열의 정렬 모드를 가져옵니다
메서드 클리어	
Clear	메모리 해제 없이 모든 배열 요소를 삭제합니다
메서드 설정	
Sort	지정된 옵션으로 배열을 정렬합니다
Input/output	
virtual Save	파일에 데이터 배열을 저장합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
virtual Load	파일에서 데이터 배열을 불러옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Step

배열의 증분 크기를 가져옵니다.

```
int  Step() const
```

Return Value

배열의 크기를 늘립니다.

예제:

```
//--- CArray::Step() 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 크기 조정 단계 가져오기
   int step=array.Step();
   //--- 배열 사용
   //--- ...
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Step

배열의 증분 크기를 설정합니다.

```
bool  Step(
   int  step      // 단계
   )
```

Parameters

step

[in] 배열의 증분 크기의 새 값.

Return Value

성공적이면 true, 0과 같거나 작은 단계를 설정하려는 시도가 있었을 경우 false.

예제:

```
//--- CArray::Step(int) 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 단계 리사이즈 설정
   bool result=array.Step(1024);
   //--- 배열 사용
   //--- ...
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Total

배열의 요소 수를 가져옵니다.

```
int  Total() const;
```

Return Value

배열에서의 요소 수.

예제:

```
//--- CArray::Total() 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 토탈 점검
   int total=array.Total();
   //--- 배열 사용
   //--- ...
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Available

추가 메모리 할당 없이 사용할 수 있는 배열의 사용 가능한 요소 수를 가져옵니다.

```
int  Available() const
```

Return Value

추가 메모리 할당 없이 사용할 수 있는 배열의 사용 가능한 요소 수.

예제:

```
//--- CArray::Available() 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 가용성 체크
   int available=array.Available();
   //--- 배열 사용
   //--- ...
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Max

메모리 재할당 없이 배열의 가능한 최대 크기를 가져옵니다.

```
int  Max() const
```

Return Value

메모리 재할당을 사용하지 않는 배열의 최대 크기.

예제:

```
//--- CArray::Max() 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 최대 크기 확인
   int max=array.Max();
   //--- 배열 사용
   //--- ...
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

IsSorted

지정된 정렬 모드를 사용하여 정렬되는 배열의 플래그를 가져옵니다.

```
bool  IsSorted(
   int  mode=0      // 정렬 모드
   ) const
```

Parameters

mode=0

[in] 테스트된 정렬 모드.

Return Value

정렬된 목록의 플래그 목록이 특정 모드를 사용하여 정렬되었다면 true, 그렇지 않으면 false.

Note

정렬 플래그를 직접 변경할 수 없습니다. Sort() 방법으로 설정되며 InserSort(...)를 제외한 모든 추가/삽입 방법으로 재설정됩니다.

예제:

```
//--- CArray::IsSorted() 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 정렬 확인
   if(array.IsSorted())
     {
      //--- 정렬 배열용 메서드 사용
      //--- ...
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SortMode

배열의 정렬 모드를 가져옵니다.

```
int  SortMode() const;
```

Return Value

정렬 모드.

예제:

```
//--- CArray::SortMode() 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 정렬 모드 확인
   int sort_mode=array.SortMode();
   //--- 배열 사용
   //--- ...
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

메모리 해제 없이 모든 배열 요소를 삭제합니다.

```
void  Clear()
```

Return Value

None.

예제:

```
//--- CArray::Clear() 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
 {
 CArray *array=new CArray;
 //---
 if(array==NULL)
 {
 printf("Object create error");
 return;
 }
 //--- 배열 사용
 //--- ...
 //--- 배열 제거
 array.Clear();
 //--- 배열 삭제
 delete array;
 }
```

© 2000-2025, MetaQuotes Ltd.

Sort

지정한 옵션을 사용하여 배열을 정렬합니다.

```
void  Sort(
   int  mode=0      // 정렬 모드
   )
```

Parameters

mode=0

[in] 배열 정렬 모드.

Return Value

No.

Note

배열 정렬은 항상 오름차순입니다. 원시 데이터 유형 배열(CArrayChar, CArrayShort 등)의 경우 'mode' 매개 변수가 사용되지 않습니다. CArrayObj 배열의 경우 다변량 정렬은 파생 클래스의 정렬(int) 방법으로 구현해야 합니다.

예제:

```
//--- CArray::Sort(int) 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- mode 0으로 정렬
   array.Sort(0);
   //--- 배열 사용
   //--- ...
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료시 true, 그렇지 않으면 false.

예제:

```
//--- CArray::Save(int) 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen() 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료시 true, 그렇지 않으면 false.

예제:

```
//--- CArray::Load(...) 예제
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayChar

CArrayChar는 char 또는 uchar 변수의 동적 배열 클래스입니다.

Description

CArrayChar 클래스는 char 또는 uchar 변수의 동적 배열로 작업할 수 있는 기능을 제공합니다. 클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

Declaration

```
class CArrayChar : public CArray
```

Title

```
#include <Arrays\ArrayChar.mqh>
```

상속 계층

CObject
  CArray
    CArrayChar

클래스 메서드

메모리 컨트롤	
Reserve	메모리를 할당하여 배열 크기를 늘립니다
Resize	배열의 새(작은) 크기를 설정합니다
Shutdown	전체 메모리 릴리스로 배열을 지웁니다
메서드 추가	
Add	배열 끝에 요소를 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
Insert	요소를 배열의 지정된 위치에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
메서드 수정	
Update	지정된 배열 위치에서 요소를 변경합니다

© 2000-2025, MetaQuotes Ltd.

メモリ コントロール	
Shift	배열의 지정된 위치에서 지정된 오프셋으로 요소를 이동합니다
메서드 삭제	
Delete	지정된 배열 위치에서 요소를 제거합니다
DeleteRange	지정된 배열 위치에서 요소 그룹을 삭제합니다
메서드 액세스	
At	지정된 배열 위치에서 요소를 가져옵니다
메서드 비교	
CompareArray	배열을 다른 배열과 비교합니다
CompareArray	배열을 다른 배열과 비교합니다
정렬된 배열 메서드	
InsertSort	정렬된 배열에 요소를 삽입합니다
Search	정렬된 배열에서 샘플과 동일한 요소를 검색합니다
SearchGreat	정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다
SearchLess	정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다
SearchGreatOrEqual	정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다
SearchLessOrEqual	정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다
SearchFirst	정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다
SearchLast	정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다
SearchLinear	배열에서 샘플과 동일한 요소를 검색합니다
Input/output	
virtual Save	데이터 배열을 파일에 저장합니다
virtual Load	파일에서 데이터 배열을 로드합니다
virtual Type	배열 유형 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

메모리를 할당하여 배열 크기를 늘립니다.

```
bool  Reserve(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 추가 요소 수.

Return Value

성공적이면 true, 0보다 작거나 혹은 같은 양을 요청하려고 했거나 배열을 늘리지 못했을 경우 false.

Note

메모리 조각화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16만큼 배열 크기를 늘립니다.

예제:

```
//--- CArrayChar::Reserve(int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 메모리 보존
   if(!array.Reserve(1024))
     {
      printf("보존 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

배열의 새(작은) 크기를 설정합니다.

```
bool  Resize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 배열의 새 크기.

Return Value

성공적이면 true, 크기를 0보다 작게 설정하려는 시도가 있을 경우 false.

Note

배열의 크기를 변경하면 메모리를 최적으로 사용할 수 있습니다. 정량보다 과도한 요소는 손실됩니다. 메모리 파편화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16에 따라 배열 크기가 변경됩니다.

예제:

```
//--- CArrayChar::Resize(int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 리사이즈
   if(!array.Resize(10))
     {
      printf("리사이즈 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

전체 메모리 릴리스로 배열을 지웁니다.

```
bool  Shutdown()
```

Return Value

성공이면 true, 오류면 false.

예제:

```
//--- CArrayChar::Shutdown() 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 셧다운
   if(!array.Shutdown())
     {
      printf("셧다운 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

배열 끝에 요소를 추가합니다.

```
bool  Add(
   char  element      // 추가할 요소
   )
```

Parameters

element

[in] 배열에 추가할 요소의 값.

Return Value

성공적이면 true, 요소를 추가할 수 없으면 false.

예제:

```
//--- CArrayChar::Add(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("요소 추가 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const char&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 추가할 소스 요소의 배열에 대한 참조.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayChar::AddArray(const char &[]) 예제
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const CArrayChar*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 추가할 요소의 소스로 사용되는 CArrayChar 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayChar::AddArray(const CArrayChar*) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete src;
      delete array;
      return;
     }
   //--- 소스 배열 삭제
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

요소를 배열의 지정된 위치에 삽입합니다.

```
bool  Insert(
   char  element,     // 삽입할 요소
   int   pos          // 위치
   )
```

Parameters

element

[in] 배열에 삽입할 요소의 값

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayChar::Insert(char,int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("삽입 실패");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   const char&  src[],     // 소스 배열
   int          pos        // 위치
   )
```

Parameters

src[]

[in] 삽입할 요소의 소스로 사용되는 배열에 대한 참조

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayChar::InsertArray(const char &[],int) 예제
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   CArrayChar*  src,      // 소스에 대한 포인터
   int           pos      // 위치
   )
```

Parameters

src

[in] 삽입할 요소의 소스로 사용되는 CArrayChar 클래스의 인스턴스에 대한 포인터.

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayChar::InsertArray(const CArrayChar*,int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const char&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 복사할 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayChar::AssignArray(const char &[]) 예제
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const CArrayChar*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 복사할 요소의 소스로 사용되는 CARrayChar 클래스의 인스턴스에 대한 포인터.

Return Value

성공시 true, 요소 복제 실패시 false.

예제:

```
//--- CArrayChar::AssignArray(const CArrayChar*) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayChar *src  =new CArrayChar;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete src;
      delete array;
      return;
     }
   //--- 배열이 동일
   //--- 소스 배열 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

지정된 배열 위치에서 요소를 변경합니다.

```
bool  Update(
   int   pos,          // 위치
   char  element       // 값
   )
```

Parameters

pos

[in] 배열에서 변경할 요소의 위치

element

[in] 요소의 새 값

Return Value

성공시 true, 요소 변경 불가시 false.

예제:

```
//--- CArrayChar::Update(int,char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 업데이트
   if(!array.Update(0,'A'))
     {
      printf("업데이트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

배열의 지정된 위치에서 지정된 간격띄우기로 항목을 이동합니다.

```
bool  Shift(
   int  pos,        // 위치
   int  shift       // 값
   )
```

Parameters

pos

[in] 배열에서 이동된 요소의 위치

shift

[in] 이동 값(양수 및 음수 모두).

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CArrayChar::Shift(int,int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 시프트
   if(!array.Shift(10,-5))
     {
      printf("시프트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

지정된 배열 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 제거할 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayChar::Delete(int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.Delete(0))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

지정된 배열 위치에서 요소 그룹을 삭제합니다.

```
bool  DeleteRange(
   int  from,     // 첫 번째 요소의 위치
   int  to        // 마지막 요소의 위치
   )
```

Parameters

from

[in] 제거할 첫 번째 배열 요소의 위치.

to

[in] 제거할 마지막 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayChar::DeleteRange(int,int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.DeleteRange(0,10))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

지정된 배열 위치에서 요소를 가져옵니다.

```
char  At(
   int  pos      // 위치
   ) const
```

Parameters

pos

[in] 배열에서 원하는 요소의 위치.

Return Value

요소의 값(success, CHAR_MAX)이 존재하지 않는 위치에서 요소를 가져오려고 했습니다(마지막 오류 코드는 ERR_OUT_OF_RANGE).

Note

물론 CHAR_MAX 는 배열 요소의 유효한 값일 수 있습니다. 따라서 이러한 값을 받은 후에는 항상 마지막 에러 코드를 확인해야 합니다.

예제:

```
//--- CArrayChar::At(int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      char result=array.At(i);
      if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 배열에서 읽는 중 오류 발생
         printf("요소 가져오기 오류");
         delete array;
         return;
        }
      //--- 요소 사용
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const char&  src[]      // 소스 배열
   ) const
```

Parameters

src[]

[in] 비교를 위한 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayChar::CompareArray(const char &[]) 예제
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const CArrayChar*  src      // 소스에 대한 포인터
   ) const
```

Parameters

src

[in] 비교를 위한 요소 소스로 사용되는 CArrayChar 클래스의 인스턴스에 대한 포인터.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayChar::CompareArray(const CArrayChar*) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

정렬된 배열에 요소를 삽입합니다.

```
bool  InsertSort(
   char  element      // 삽입할 요소
   )
```

Parameters

element

[in] 정렬된 배열에 삽입할 요소의 값

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayChar::InsertSort(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 삽입
   if(!array.InsertSort('A'))
     {
      printf("삽입 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 배열에서 샘플과 같은 요소를 검색합니다.

```
int  Search(
   char  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayChar::Search(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.Search('A')!=-1) printf("요소가 검색됨");
   else                      printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다.

```
int  SearchGreat(
   char  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

성공적이면 검색된 요소의 위치, 요소가 검색되지 않으면 -1.

예제:

```
//--- CArrayChar::SearchGreat(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreat('A')!=-1) printf("요소가 검색됨");
   else                           printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다.

```
int  SearchLess(
   char  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayChar::SearchLess(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLess('A')!=-1) printf("요소가 검색됨");
   else                          printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchGreatOrEqual(
   char  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayChar::SearchGreatOrEqual(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreatOrEqual('A')!=-1) printf("요소가 검색됨");
   else                                  printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchLessOrEqual(
   char  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayChar::SearchLessOrEqual(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLessOrEqual('A')!=-1) printf("요소가 검색됨");
   else                                 printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다.

```
int  SearchFirst(
   char  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayChar::SearchFirst(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchFirst('A')!=-1) printf("요소가 검색됨");
   else                           printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다.

```
int  SearchLast(
   char  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayChar::SearchLast(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLast('A')!=-1) printf("요소가 검색됨");
   else                          printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

배열에서 샘플과 동일한 요소를 검색합니다.

```
int  SearchLinear(
   char  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

Note

이 메소드는 정렬되지 않은 배열에 선형 검색(또는 순차 검색) 알고리즘을 사용합니다.

예제:

```
//--- CArrayChar::SearchLinear(char) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 검색
   if(array.SearchLinear('A')!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayChar::Save(int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayChar::Load(int) 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 요소 사용
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%c'",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

배열 유형 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

배열 유형 식별자(CArrayChar -77용).

예제:

```
//--- CArrayChar::Type() 예제
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 유형 가져오기
   int type=array.Type();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayShort

CArrayShort는 동적 배열의 short 또는 ushort 변수 클래스입니다.

Description

Class CArrayShort는 short 또는 ushort 변수의 동적 배열로 작업할 수 있는 기능을 제공합니다. 클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

Declaration

```
class CArrayShort : public CArray
```

Title

```
#include <Arrays\ArrayShort.mqh>
```

상속 계층

CObject
  CArray
    CArrayShort

그룹별 클래스 메서드

메모리 컨트롤	
Reserve	메모리를 할당하여 배열 크기를 늘립니다
Resize	배열의 새(작은) 크기를 설정합니다
Shutdown	전체 메모리 릴리스로 배열을 지웁니다
메서드 추가	
Add	배열 끝에 요소를 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
Insert	요소를 배열의 지정된 위치에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
메서드 업데이트	
Update	지정된 배열 위치에서 요소를 변경합니다

© 2000-2025, MetaQuotes Ltd.

메모리 컨트롤	
Shift	배열의 지정된 위치에서 지정된 오프셋으로 요소를 이동합니다
메서드 삭제	
Delete	지정된 배열 위치에서 요소를 제거합니다
DeleteRange	지정된 배열 위치에서 요소 그룹을 삭제합니다
메서드 액세스	
At	지정된 배열 위치에서 요소를 가져옵니다
메서드 비교	
CompareArray	배열을 다른 배열과 비교합니다
CompareArray	배열을 다른 배열과 비교합니다
정렬된 배열 작업	
InsertSort	정렬된 배열에 요소를 삽입합니다
Search	정렬된 배열에서 샘플과 동일한 요소를 검색합니다
SearchGreat	정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다
SearchLess	정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다
SearchGreatOrEqual	정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다
SearchLessOrEqual	정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다
SearchFirst	정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다
SearchLast	정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다
SearchLinear	배열에서 샘플과 동일한 요소를 검색합니다
Input/output	
virtual Save	데이터 배열을 파일에 저장합니다
virtual Load	파일에서 데이터 배열을 로드합니다
virtual Type	배열의 유형 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

메모리를 할당하여 배열 크기를 늘립니다.

```
bool  Reserve(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 추가 요소 수.

Return Value

성공적이면 true, 0보다 작거나 혹은 같은 양을 요청하려고 했거나 배열을 늘리지 못했을 경우 false.

Note

메모리 조각화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16을 사용하여 배열 크기를 변경합니다.

예제:

```
//--- CArrayShort::Reserve(int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 메모리 보존
   if(!array.Reserve(1024))
     {
      printf("보존 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

배열의 새(작은) 크기를 설정합니다.

```
bool  Resize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 배열의 새 크기.

Return Value

성공적이면 true, 크기를 0보다 작게 설정하려는 시도가 있을 경우 false.

Note

배열의 크기를 변경하면 메모리를 최적으로 사용할 수 있습니다. 정량보다 과도한 요소는 손실됩니다. 메모리 파편화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16에 따라 배열 크기가 변경됩니다.

예제:

```
//--- CArrayShort::Resize(int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 리사이즈
   if(!array.Resize(10))
     {
      printf("리사이즈 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

전체 메모리 릴리스로 배열을 지웁니다.

```
bool  Shutdown()
```

Return Value

성공이면 true, 오류면 false.

예제:

```
//--- CArrayShort::Shutdown() 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 셧다운
   if(!array.Shutdown())
     {
      printf("셧다운 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

배열 끝에 요소를 추가합니다.

```
bool  Add(
   short  element      // 추가할 요소
   )
```

Parameters

element

[in] 배열에 추가할 요소의 값.

Return Value

성공적이면 true, 요소를 추가할 수 없으면 false.

예제:

```
//--- CArrayShort::Add(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("요소 추가 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const short&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 추가할 소스 요소의 배열에 대한 참조.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayShort::AddArray(const short &[]) 예제
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const CArrayShort*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 추가할 요소의 소스로 사용되는 CArrayShort 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayShort::AddArray(const CArrayShort*) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete src;
      delete array;
      return;
     }
   //--- 소스 배열 삭제
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

요소를 배열의 지정된 위치에 삽입합니다.

```
bool  Insert(
   short  element,     // 삽입할 요소
   int    pos          // 위치
   )
```

Parameters

element

[in] 배열에 삽입할 요소의 값

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayShort::Insert(short,int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("삽입 실패");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   const short&  src[],     // 소스 배열
   int           pos        // 포지션
   )
```

Parameters

src[]

[in] 삽입할 요소의 소스로 사용되는 배열에 대한 참조

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayShort::InsertArray(const short &[],int) 예제
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   CArrayShort*  src,     // 소스에 대한 포인터
   int           pos      // 위치
   )
```

Parameters

src

[in] 삽입할 요소의 소스로 사용되는 CArrayShort 클래스의 인스턴스에 대한 포인터.

pos

[in] 삽입할 배열의 위치.

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayShort::InsertArray(const CArrayShort*,int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const short&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 복사할 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayShort::AssignArray(const short &[]) 예제
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const CArrayShort*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 복사할 요소의 소스로 사용되는 CArrayShort 클래스의 인스턴스에 대한 포인터.

Return Value

성공시 true, 요소 복제 실패시 false.

예제:

```
//--- CArrayShort::AssignArray(const CArrayShort*) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayShort *src  =new CArrayShort;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete src;
      delete array;
      return;
     }
   //--- 어레이가 동일
   //--- 소스 배열 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

지정된 배열 위치에서 요소를 변경합니다.

```
bool  Update(
   int    pos,          // 위치
   short  element       // 값
   )
```

Parameters

pos

[in] 배열에서 변경할 요소의 위치

element

[in] 요소의 새 값

Return Value

성공시 true, 요소 변경 불가시 false.

예제:

```
//--- CArrayShort::Update(int,short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 업데이트
   if(!array.Update(0,100))
     {
      printf("업데이트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

배열의 지정된 위치에서 지정된 간격띄우기로 항목을 이동합니다.

```
bool  Shift(
   int  pos,        // 위치
   int  shift       // 시프트
   )
```

Parameters

pos

[in] 배열에서 이동된 요소의 위치

shift

[in] 이동 값(양수 및 음수 모두).

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CArrayShort::Shift(int,int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 시프트
   if(!array.Shift(10,-5))
     {
      printf("시프트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

지정된 배열 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 제거할 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayShort::Delete(int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.Delete(0))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

지정된 배열 위치에서 요소 그룹을 삭제합니다.

```
bool  DeleteRange(
   int  from,     // 첫 번째 요소의 위치
   int  to        // 마지막 요소의 위치
   )
```

Parameters

from

[in] 제거할 첫 번째 배열 요소의 위치.

to

[in] 제거할 마지막 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayShort::DeleteRange(int,int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.DeleteRange(0,10))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

지정된 배열 위치에서 요소를 가져옵니다.

```
short  At(
   int  pos      // 위치
   ) const
```

Parameters

pos

[in] 배열에서 원하는 요소의 위치.

Return Value

성공시 요소의 값, 존재하지 않는 위치에서 요소를 가져오려는 시도가 있었을 경우(최근 오류 코드가 ERR_OUT_OF_RANGE) - SHORT_MAX.

Note

물론 SHORT_MAX는 배열 요소의 유효한 값일 수 있습니다. 따라서 이러한 값을 받은 후에는 항상 마지막 에러 코드를 확인해야 합니다.

예제:

```
//--- CArrayShort::At(int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      short result=array.At(i);
      if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 배열에서 읽는 중 오류 발생
         printf("요소 가져오기 오류");
         delete array;
         return;
        }
      //--- 요소 사용
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const short&  src[]      // 소스 배열
   ) const
```

Parameters

src[]

[in] 비교를 위한 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayShort::CompareArray(const short &[]) 예제
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const CArrayShort*  src      // 소스에 대한 포인터
   ) const
```

Parameters

src

[in] 비교를 위한 요소 소스로 사용되는 CArrayShort 클래스의 인스턴스에 대한 포인터.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayShort::CompareArray(const CArrayShort*) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

정렬된 배열에 요소를 삽입합니다.

```
bool  InsertSort(
   short  element      // 삽입할 요소
   )
```

Parameters

element

[in] 정렬된 배열에 삽입할 요소의 값

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayShort::InsertSort(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 삽입
   if(!array.InsertSort(100))
     {
      printf("삽입 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 배열에서 샘플과 같은 요소를 검색합니다.

```
int  Search(
   short  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayShort::Search(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.Search(100)!=-1) printf("요소가 검색됨");
   else                      printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다.

```
int  SearchGreat(
   short  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

성공적이면 -검색된 요소의 위치, 검색되지 않으면 -1.

예제:

```
//--- CArrayShort::SearchGreat(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreat(100)!=-1) printf("요소가 검색됨");
   else                           printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다.

```
int  SearchLess(
   short  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayShort::SearchLess(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLess(100)!=-1) printf("요소가 검색됨");
   else                          printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchGreatOrEqual(
   short  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayShort::SearchGreatOrEqual(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreatOrEqual(100)!=-1) printf("요소가 검색됨");
   else                                  printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchLessOrEqual(
   short  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayShort::SearchLessOrEqual(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLessOrEqual(100)!=-1) printf("요소가 검색됨");
   else                                 printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다.

```
int  SearchFirst(
   short  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayShort::SearchFirst(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchFirst(100)!=-1) printf("요소가 검색됨");
   else                           printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다.

```
int  SearchLast(
   short  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayShort::SearchLast(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLast(100)!=-1) printf("요소가 검색됨");
   else                          printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

배열에서 표본과 같은 요소를 검색합니다.

```
int  SearchLinear(
   short  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

Note

이 메소드는 정렬되지 않은 배열에 선형 검색(또는 순차 검색) 알고리즘을 사용합니다.

예제:

```
//--- CArrayShort::SearchLinear(short) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 검색
   if(array.SearchLinear(100)!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayShort::Save(int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 100 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayShort::Load(int) 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 요소 사용
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

배열 유형 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

배열 유형 식별자(CArrayShort -82용).

예제:

```
//--- CArrayShort::Type() 예제
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 유형 가져오기
   int type=array.Type();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayInt

CArrayInt는 int 또는 unit 변수의 동적 배열 클래스입니다.

Description

CArrayInt 클래스는 int 또는 uint 변수의 동적 배열로 작업할 수 있는 기능을 제공합니다. 클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

Declaration

```
class CArrayInt : public CArray
```

Title

```
#include <Arrays\ArrayInt.mqh>
```

상속 계층

CObject
  CArray
    CArrayInt

직계 종속

CSpreadBuffer

그룹별 클래스 메서드

메모리 컨트롤	
Reserve	메모리를 할당하여 배열 크기를 늘립니다
Resize	배열의 새(작은) 크기를 설정합니다
Shutdown	전체 메모리 릴리스로 배열을 지웁니다
메서드 추가	
Add	배열 끝에 요소를 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
Insert	요소를 배열의 지정된 위치에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다

© 2000-2025, MetaQuotes Ltd.

메모리 컨트롤	
메서드 업데이트	
Update	지정된 배열 위치에서 요소를 변경합니다
Shift	배열의 지정된 위치에서 지정된 오프셋으로 요소를 이동합니다
메서드 삭제	
Delete	지정된 배열 위치에서 요소를 제거합니다
DeleteRange	지정된 배열 위치에서 요소 그룹을 삭제합니다
메서드 액세스	
At	지정된 배열 위치에서 요소를 가져옵니다
메서드 비교	
CompareArray	배열을 다른 배열과 비교합니다
CompareArray	배열을 다른 배열과 비교합니다
정렬된 배열 작업	
InsertSort	정렬된 배열에 요소를 삽입합니다
Search	정렬된 배열에서 샘플과 동일한 요소를 검색합니다
SearchGreat	정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다
SearchLess	정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다
SearchGreatOrEqual	정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다
SearchLessOrEqual	정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다
SearchFirst	정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다
SearchLast	정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다
SearchLinear	배열에서 샘플과 동일한 요소를 검색합니다
Input/output	
virtual Save	데이터 배열을 파일에 저장합니다
virtual Load	파일에서 데이터 배열을 로드합니다
virtual Type	배열의 유형 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

메모리를 할당하여 배열 크기를 늘립니다.

```
bool  Reserve(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 추가 요소 수.

Return Value

성공적이면 true, 0보다 작거나 혹은 같은 양을 요청하려고 했거나 배열을 늘리지 못했을 경우 false.

Note

메모리 조각화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16을 사용하여 배열 크기를 변경합니다.

예제:

```
//--- CArrayInt::Reserve(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 메모리 보존
   if(!array.Reserve(1024))
     {
      printf("보존 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

배열의 새(작은) 크기를 설정합니다.

```
bool  Resize(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 새 크기.

Return Value

성공적이면 true, 크기를 0보다 작게 설정하려는 시도가 있을 경우 false.

Note

배열의 크기를 변경하면 메모리를 최적으로 사용할 수 있습니다. 정량보다 과도한 요소는 손실됩니다. 메모리 파편화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16에 따라 배열 크기가 변경됩니다.

예제:

```
//--- CArrayInt::Resize(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 리사이즈
   if(!array.Resize(10))
     {
      printf("리사이즈 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

전체 메모리 릴리스로 배열을 지웁니다.

```
bool  Shutdown()
```

Return Value

성공이면 true, 오류면 false.

예제:

```
//--- CArrayInt::Shutdown() 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 셧다운
   if(!array.Shutdown())
     {
      printf("셧다운 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

배열 끝에 요소를 추가합니다.

```
bool  Add(
   int  element      // 추가할 요소
   )
```

Parameters

element

[in] 배열에 추가할 요소의 값.

Return Value

성공적이면 true, 요소를 추가할 수 없으면 false.

예제:

```
//--- CArrayInt::Add(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("요소 추가 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const int&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 추가할 소스 요소의 배열에 대한 참조.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayInt::AddArray(const int &[]) 예제
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const CArrayInt*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 추가할 요소의 원본으로 사용되는 CArrayInt 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayInt::AddArray(const CArrayInt*) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete src;
      delete array;
      return;
     }
   //--- 소스 배열 삭제
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

요소를 배열의 지정된 위치에 삽입합니다.

```
bool  Insert(
   int  element,     // 삽입할 요소
   int  pos          // 위치
   )
```

Parameters

element

[in] 배열에 삽입할 요소의 값

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayInt::Insert(int,int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("삽입 실패");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   const int&  src[],     // 소스 배열
   int         pos        // 위치
   )
```

Parameters

src[]

[in] 삽입할 요소의 소스로 사용되는 배열에 대한 참조

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayInt::InsertArray(const int &[],int) 예제
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   CArrayInt*  src,     // 소스에 대한 포인터
   int          pos      // 위치
   )
```

Parameters

src

[in] 삽입할 요소의 원본으로 사용되는 CArrayInt 클래스의 인스턴스에 대한 포인터.

pos

[in] 삽입할 배열의 위치.

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayInt::InsertArray(const CArrayInt*,int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const int&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 복사할 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayInt::AssignArray(const int &[]) 예제
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const CArrayInt*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 복사할 요소의 소스로 사용되는 CArrayInt 클래스의 인스턴스에 대한 포인터.

Return Value

성공시 true, 요소 복제 실패시 false.

예제:

```
//--- CArrayInt::AssignArray(const CArrayInt*) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayInt *src  =new CArrayInt;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete src;
      delete array;
      return;
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 어레이가 동일
   //--- 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

지정된 배열 위치에서 요소를 변경합니다.

```
bool  Update(
   int  pos,          // 위치
   int  element       // 값
   )
```

Parameters

pos

[in] 배열에서 변경할 요소의 위치.

element

[in] 요소의 새 값

Return Value

성공시 true, 요소 변경 불가시 false.

예제:

```
//--- CArrayInt::Update(int,int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 업데이트
   if(!array.Update(0,10000))
     {
      printf("업데이트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

배열의 지정된 위치에서 지정된 간격띄우기로 항목을 이동합니다.

```
bool  Shift(
   int  pos,        // 위치
   int  shift       // 시프트
   )
```

Parameters

pos

[in] 배열에서 이동된 요소의 위치

shift

[in] 이동 값(양수 및 음수 모두).

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CArrayInt::Shift(int,int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 시프트
   if(!array.Shift(10,-5))
     {
      printf("시프트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

지정된 배열 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 제거할 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayInt::Delete(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.Delete(0))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

지정된 배열 위치에서 요소 그룹을 삭제합니다.

```
bool  DeleteRange(
   int  from,     // 첫 번째 요소의 위치
   int  to        // 마지막 요소의 위치
   )
```

Parameters

from

[in] 제거할 첫 번째 배열 요소의 위치.

to

[in] 제거할 마지막 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayInt::DeleteRange(int,int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.DeleteRange(0,10))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

지정된 배열 위치에서 요소를 가져옵니다.

```
int  At(
   int  pos      // 위치
   ) const
```

Parameters

pos

[in] 배열에서 원하는 요소의 위치.

Return Value

성공시 -요소 값, 존재하지 않는 위치에서 요소를 가져오려는 시도가 있을 경우(최근 마지막 오류 코드가 ERR_OUT_OF_RANGE) -INT_MAX.

Note

물론 INT_MAX는 배열 요소의 유효한 값일 수 있습니다. 따라서 이러한 값을 받은 후에는 항상 마지막 에러 코드를 확인해야 합니다.

예제:

```
//--- CArrayInt::At(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      int result=array.At(i);
      if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 배열에서 읽는 중 오류 발생
         printf("요소 가져오기 오류");
         delete array;
         return;
        }
      //--- 요소 사용
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const int&  src[]      // 소스 배열
   ) const
```

Parameters

src[]

[in] 비교를 위한 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayInt::CompareArray(const int &[]) 예제
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const CArrayInt*  src      // 소스에 대한 포인터
   ) const
```

Parameters

src

[in] 비교를 위한 요소 소스로 사용되는 CArrayInt 클래스의 인스턴스에 대한 포인터.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayInt::CompareArray(const CArrayInt*) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

정렬된 배열에 요소를 삽입합니다.

```
bool  InsertSort(
   int  element      // 삽입할 요소
   )
```

Parameters

element

[in] 정렬된 배열에 삽입할 요소의 값

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayInt::InsertSort(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 삽입
   if(!array.InsertSort(10000))
     {
      printf("삽입 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 배열에서 샘플과 같은 요소를 검색합니다.

```
int  Search(
   int  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayInt::Search(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.Search(10000)!=-1) printf("요소가 검색됨");
   else                        printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다.

```
int  SearchGreat(
   int  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayInt::SearchGreat(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreat(10000)!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다.

```
int  SearchLess(
   int  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayInt::SearchLess(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLess(10000)!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchGreatOrEqual(
   int  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayInt::SearchGreatOrEqual(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreatOrEqual(10000)!=-1) printf("요소가 검색됨");
   else                                    printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchLessOrEqual(
   int  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayInt::SearchLessOrEqual(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLessOrEqual(10000)!=-1) printf("요소가 검색됨");
   else                                   printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다.

```
int  SearchFirst(
   int  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayInt::SearchFirst(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchFirst(10000)!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다.

```
int  SearchLast(
   int  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayInt::SearchLast(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLast(10000)!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

배열에서 샘플과 동일한 요소를 검색합니다.

```
int  SearchLinear(
   int  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

Note

이 메소드는 정렬되지 않은 배열에 선형 검색(또는 순차 검색) 알고리즘을 사용합니다.

예제:

```
//--- CArrayInt::SearchLinear(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 검색
   if(array.SearchLinear(10000)!=-1) printf("요소가 검색됨");
   else                              printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayInt::Save(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 100 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayInt::Load(int) 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 요소 사용
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

배열 유형 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

Array type identifier (for CArrayInt - 82).

예제:

```
//--- CArrayInt::Type() 예제
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 유형 가져오기
   int type=array.Type();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayLong

CARayLong 클래스는 long 변수 또는 ulong 변수의 동적 배열 클래스입니다.

Description

CArrayLong 클래스는 long 변수 또는 ulong 변수의 동적 배열로 작업할 수 있는 기능을 제공합니다. 클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

Declaration

```
class CArrayLong : public CArray
```

Title

```
#include <Arrays\ArrayLong.mqh>
```

상속 계층

CObject
  CArray
    CArrayLong

직계 종속

CRealVolumeBuffer, CTickVolumeBuffer, CTimeBuffer

그룹별 클래스 메서드

메모리 컨트롤	
Reserve	메모리를 할당하여 배열 크기를 늘립니다
Resize	배열의 새(작은) 크기를 설정합니다
Shutdown	전체 메모리 릴리스로 배열을 지웁니다
메서드 추가	
Add	배열 끝에 요소를 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
Insert	요소를 배열의 지정된 위치에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다

© 2000-2025, MetaQuotes Ltd.

메모리 컨트롤	
메서드 업데이트	
Update	지정된 배열 위치에서 요소를 변경합니다
Shift	배열의 지정된 위치에서 지정된 오프셋으로 요소를 이동합니다
메서드 삭제	
Delete	지정된 배열 위치에서 요소를 제거합니다
DeleteRange	지정된 배열 위치에서 요소 그룹을 삭제합니다
메서드 액세스	
At	지정된 배열 위치에서 요소를 가져옵니다
메서드 비교	
CompareArray	배열을 다른 배열과 비교합니다
CompareArray	배열을 다른 배열과 비교합니다
정렬된 배열 작업	
InsertSort	정렬된 배열에 요소를 삽입합니다
Search	정렬된 배열에서 샘플과 동일한 요소를 검색합니다
SearchGreat	정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다
SearchLess	정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다
SearchGreatOrEqual	정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다
SearchLessOrEqual	정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다
SearchFirst	정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다
SearchLast	정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다
SearchLinear	배열에서 샘플과 동일한 요소를 검색합니다
Input/output	
virtual Save	데이터 배열을 파일에 저장합니다
virtual Load	파일에서 데이터 배열을 로드합니다
virtual Type	배열의 유형 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

메모리를 할당하여 배열 크기를 늘립니다.

```
bool  Reserve(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 추가 요소 수.

Return Value

성공적이면 true, 0보다 작거나 혹은 같은 양을 요청하려고 했거나 배열을 늘리지 못했을 경우 false.

Note

메모리 조각화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16을 사용하여 배열 크기를 변경합니다.

예제:

```
//--- CArrayLong::Reserve(int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 메모리 보존
   if(!array.Reserve(1024))
     {
      printf("보존 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

배열의 새(작은) 크기를 설정합니다.

```
bool  Resize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 배열의 새 크기.

Return Value

성공적이면 true, 크기를 0보다 작게 설정하려는 시도가 있을 경우 false.

Note

배열의 크기를 변경하면 메모리를 최적으로 사용할 수 있습니다. 정량보다 과도한 요소는 손실됩니다. 메모리 파편화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16에 따라 배열 크기가 변경됩니다.

예제:

```
//--- CArrayLong::Resize(int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 리사이즈
   if(!array.Resize(10))
     {
      printf("리사이즈 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

전체 메모리 릴리스로 배열을 지웁니다.

```
bool  Shutdown()
```

Return Value

성공이면 true, 오류면 false.

예제:

```
//--- CArrayLong::Shutdown() 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 셧다운
   if(!array.Shutdown())
     {
      printf("셧다운 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

배열 끝에 요소를 추가합니다.

```
bool  Add(
   long  element      // 추가할 요소
   )
```

Parameters

element

[in] 배열에 추가할 요소의 값.

Return Value

성공적이면 true, 요소를 추가할 수 없으면 false.

예제:

```
//--- CArrayLong::Add(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("요소 추가 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const long&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 추가할 소스 요소의 배열에 대한 참조.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayLong::AddArray(const long &[]) 예제
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const CArrayLong*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 추가할 요소의 소스로 사용되는 CArrayLong 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayLong::AddArray(const CArrayLong*) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete src;
      delete array;
      return;
     }
   //--- 소스 배열 삭제
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

요소를 배열의 지정된 위치에 삽입합니다.

```
bool  Insert(
   long  element,     // 삽입할 요소
   int   pos          // 위치
   )
```

Parameters

element

[in] 배열에 삽입할 요소의 값

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayLong::Insert(long,int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("삽입 실패");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   const long&  src[],     // 소스 배열
   int          pos        // 위치
   )
```

Parameters

src[]

[in] 삽입할 요소의 소스로 사용되는 배열에 대한 참조

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayLong::InsertArray(const long &[],int) 예제
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   CArrayLong*  src,      // 소스에 대한 포인터
   int           pos      // 위치
   )
```

Parameters

src

[in] 삽입할 요소의 소스로 사용되는 CArrayLong 클래스의 인스턴스에 대한 포인터.

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayLong::InsertArray(const CArrayLong*,int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const long&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 복사할 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayLong::AssignArray(const long &[]) 예제
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const CArrayLong*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 복사할 요소의 소스로 사용되는 CArrayLong 클래스의 인스턴스에 대한 포인터.

Return Value

성공시 true, 요소 복제 실패시 false.

예제:

```
//--- CArrayLong::AssignArray(const CArrayLong*) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayLong *src  =new CArrayLong;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete src;
      delete array;
      return;
     }
   //--- 어레이가 동일
   //--- 소스 배열 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

지정된 배열 위치에서 요소를 변경합니다.

```
bool  Update(
   int   pos,          // 위치
   long  element       // 값
   )
```

Parameters

pos

[in] 배열에서 변경할 요소의 위치

element

[in] 요소의 새 값

Return Value

성공시 true, 요소 변경 불가시 false.

예제:

```
//--- CArrayLong::Update(int,long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 업데이트
   if(!array.Update(0,1000000))
     {
      printf("업데이트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

배열의 지정된 위치에서 지정된 간격띄우기로 항목을 이동합니다.

```
bool  Shift(
   int  pos,        // 위치
   int  shift       // 시프트
   )
```

Parameters

pos

[in] 배열에서 이동된 요소의 위치

shift

[in] 이동 값(양수 및 음수 모두).

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CArrayLong::Shift(int,int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 시프트
   if(!array.Shift(10,-5))
     {
      printf("시프트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

지정된 배열 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 제거할 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayLong::Delete(int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.Delete(0))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

지정된 배열 위치에서 요소 그룹을 삭제합니다.

```
bool  DeleteRange(
   int  from,     // 첫 번째 요소의 위치
   int  to        // 마지막 요소의 위치
   )
```

Parameters

from

[in] 제거할 첫 번째 배열 요소의 위치.

to

[in] 제거할 마지막 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayLong::DeleteRange(int,int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.DeleteRange(0,10))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

지정된 배열 위치에서 요소를 가져옵니다.

```
long  At(
   int  pos      // 위치
   ) const
```

Parameters

pos

[in] 배열에서 원하는 요소의 위치.

Return Value

성공시 – 요소의 값, 존재하지 않는 위치에서 요소를 가져오려는 시도가 있을 경우(최근 오류 코드가 ERR_OUT_OF-RANGE) - LONG_MAX.

Note

물론 LONG_MAX는 배열 요소의 유효한 값일 수 있습니다. 따라서 이러한 값을 받은 후에는 항상 마지막 에러 코드를 확인해야 합니다.

예제:

```
//--- CArrayLong::At(int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      long result=array.At(i);
      if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 배열에서 읽는 중 오류 발생
         printf("요소 가져오기 오류");
         delete array;
         return;
        }
      //--- 요소 사용
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const long&  src[]      // 소스 배열
   ) const
```

Parameters

src[]

[in] 비교를 위한 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayLong::CompareArray(const long &[]) 예제
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArrayconst

배열을 다른 배열과 비교합니다.

```
bool  CompareArrayconst(
   const CArrayLong*  src      // 소스에 대한 포인터
   ) const
```

Parameters

src

[in] 비교를 위한 요소 소스로 사용되는 CArrayLong 클래스의 인스턴스에 대한 포인터.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayLong::CompareArray(const CArrayLong*) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

정렬된 배열에 요소를 삽입합니다.

```
bool  InsertSort(
   long  element      // 삽입할 요소
   )
```

Parameters

element

[in] 정렬된 배열에 삽입할 요소의 값

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayLong::InsertSort(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 삽입
   if(!array.InsertSort(1000000))
     {
      printf("삽입 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 배열에서 샘플과 같은 요소를 검색합니다.

```
int  Search(
   long  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayLong::Search(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.Search(1000000)!=-1) printf("요소가 검색됨");
   else                          printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다.

```
int  SearchGreat(
   long  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayLong::SearchGreat(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreat(1000000)!=-1) printf("요소가 검색됨");
   else                               printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다.

```
int  SearchLess(
   long  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayLong::SearchLess(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLess(1000000)!=-1) printf("요소가 검색됨");
   else                              printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchGreatOrEqual(
   long  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayLong::SearchGreatOrEqual(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreatOrEqual(1000000)!=-1) printf("요소가 검색됨");
   else                                      printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchLessOrEqual(
   long  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayLong::SearchLessOrEqual(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLessOrEqual(1000000)!=-1) printf("요소가 검색됨");
   else                                     printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다.

```
int  SearchFirst(
   long  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayLong::SearchFirst(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchFirst(1000000)!=-1) printf("요소가 검색됨");
   else                               printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다.

```
int  SearchLast(
   long  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayLong::SearchLast(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLast(1000000)!=-1) printf("요소가 검색됨");
   else                              printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

배열에서 샘플과 동일한 요소를 검색합니다.

```
int  SearchLinear(
   long  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

Note

이 메소드는 정렬되지 않은 배열에 선형 검색(또는 순차 검색) 알고리즘을 사용합니다.

예제:

```
//--- CArrayLong::SearchLinear(long) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 검색
   if(array.SearchLinear(1000000)!=-1) printf("요소가 검색됨");
   else                                printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayLong::Save(int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 100 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayLong::Load(int) 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 요소 사용
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %I64",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

배열 유형 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

배열 유형 식별자(CArrayLong -84용).

예제:

```
//--- CArrayLong::Type() 예제
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 유형 가져오기
   int type=array.Type();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayFloat

CArrayFloat 클래스는 부동 변수의 동적 배열 클래스입니다.

Description

CArrayFloat 클래스는 유동 변수의 동적 배열로 작업할 수 있는 기능을 제공합니다. 클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

Declaration

```
class CArrayFloat : public CArray
```

Title

```
#include <Arrays\ArrayFloat.mqh>
```

상속 계층

CObject
  CArray
    CArrayFloat

그룹별 클래스 메서드

Attributes	
Delta	비교 허용오차를 설정합니다.
메모리 컨트롤	
Reserve	메모리를 할당하여 배열 크기를 늘립니다
Resize	배열의 새(작은) 크기를 설정합니다
Shutdown	전체 메모리 릴리스로 배열을 지웁니다
메서드 추가	
Add	배열 끝에 요소를 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
Insert	요소를 배열의 지정된 위치에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
메서드 업데이트	
Update	지정된 배열 위치에서 요소를 변경합니다
Shift	배열의 지정된 위치에서 지정된 오프셋으로 요소를 이동합니다
메서드 삭제	
Delete	지정된 배열 위치에서 요소를 제거합니다
DeleteRange	지정된 배열 위치에서 요소 그룹을 삭제합니다
메서드 액세스	
At	지정된 배열 위치에서 요소를 가져옵니다
메서드 비교	
CompareArray	배열을 다른 배열과 비교합니다
CompareArray	배열을 다른 배열과 비교합니다
정렬된 배열 작업	
InsertSort	정렬된 배열에 요소를 삽입합니다
Search	정렬된 배열에서 샘플과 동일한 요소를 검색합니다
SearchGreat	정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다
SearchLess	정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다
SearchGreatOrEqual	정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다
SearchLessOrEqual	정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다
SearchFirst	정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다
SearchLast	정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다
SearchLinear	배열에서 샘플과 동일한 요소를 검색합니다
Input/output	
virtual Save	데이터 배열을 파일에 저장합니다
virtual Load	파일에서 데이터 배열을 로드합니다
virtual Type	배열의 유형 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Delta

비교 허용오차를 설정합니다.

```
void  Delta(
   float  delta      // 허용 오차
   )
```

Parameters

delta

[in] 비교 허용오차의 새 값.

Return Value

None

Note

비교 허용 오차는 검색에 사용됩니다. 값의 차이가 허용오차보다 작거나 같으면 같은 것으로 간주됩니다. 기본 허용오차 값은 0.0입니다.

예제:

```
//--- CArrayFloat::Delta(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 비교 변화량 설정
   array.Delta(0.001);
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

메모리를 할당하여 배열 크기를 늘립니다.

```
bool  Reserve(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 추가 요소 수.

Return Value

성공적이면 true, 0보다 작거나 혹은 같은 양을 요청하려고 했거나 배열을 늘리지 못했을 경우 false.

Note

메모리 조각화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16을 사용하여 배열 크기를 변경합니다.

예제:

```
//--- CArrayFloat::Reserve(int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 메모리 보존
   if(!array.Reserve(1024))
     {
      printf("보존 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

배열의 새(작은) 크기를 설정합니다.

```
bool  Resize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 배열의 새 크기.

Return Value

성공적이면 true, 크기를 0보다 작게 설정하려는 시도가 있을 경우 false.

Note

배열의 크기를 변경하면 메모리를 최적으로 사용할 수 있습니다. 정량보다 과도한 요소는 손실됩니다. 메모리 파편화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16에 따라 배열 크기가 변경됩니다.

예제:

```
//--- CArrayFloat::Resize(int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 리사이즈
   if(!array.Resize(10))
     {
      printf("리사이즈 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

전체 메모리 릴리스로 배열을 지웁니다.

```
bool  Shutdown()
```

Return Value

성공이면 true, 오류면 false.

예제:

```
//--- CArrayFloat::Shutdown() 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 셧다운
   if(!array.Shutdown())
     {
      printf("셧다운 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

배열 끝에 요소를 추가합니다.

```
bool  Add(
   float  element      // 추가할 요소
   )
```

Parameters

element

[in] 배열에 추가할 요소의 값.

Return Value

성공적이면 true, 요소를 추가할 수 없으면 false.

예제:

```
//--- CArrayFloat::Add(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("요소 추가 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const float&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 추가할 소스 요소의 배열에 대한 참조.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayFloat::AddArray(const float &[]) 예제
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const CArrayFloat*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 추가할 요소의 소스로 사용되는 CArrayFloat 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayFloat::AddArray(const CArrayFloat*) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete src;
      delete array;
      return;
     }
   //--- 소스 배열 삭제
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

요소를 배열의 지정된 위치에 삽입합니다.

```
bool  Insert(
   float  element,     // 삽입할 요소
   int    pos          // 위치
   )
```

Parameters

element

[in] 배열에 삽입할 요소의 값

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayFloat::Insert(float,int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("삽입 실패");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   const float&  src[],     // 소스 배열
   int           pos        // 포지션
   )
```

Parameters

src[]

[in] 삽입할 요소의 소스로 사용되는 배열에 대한 참조

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayFloat::InsertArray(const float &[],int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   CArrayFloat*  src,      // 소스에 대한 포인터
   int           pos       // 위치
   )
```

Parameters

src

[in] 삽입할 요소의 소스로 사용되는 CArrayFloat 클래스의 인스턴스에 대한 포인터.

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayFloat::InsertArray(const CArrayFloat*,int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const float&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 복사할 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayFloat::AssignArray(const float &[]) 예제
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const CArrayFloat*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 복사할 요소의 소스로 사용되는 CArrayFloat 클래스의 인스턴스에 대한 포인터.

Return Value

성공시 true, 요소 복제 실패시 false.

예제:

```
//--- CArrayFloat::AssignArray(const CArrayFloat*) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayFloat *src  =new CArrayFloat;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete src;
      delete array;
      return;
     }
   //--- 어레이가 동일
   //--- 소스 배열 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

지정된 배열 위치에서 요소를 변경합니다.

```
bool  Update(
   int    pos,          // 위치
   float  element       // 값
   )
```

Parameters

pos

[in] 배열에서 변경할 요소의 위치

element

[in] 요소의 새 값

Return Value

성공시 true, 요소 변경 불가시 false.

예제:

```
//--- CArrayFloat::Update(int,float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 업데이트
   if(!array.Update(0,100.0))
     {
      printf("업데이트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

배열의 지정된 위치에서 지정된 오프셋으로 항목을 이동합니다.

```
bool  Shift(
   int  pos,        // 위치
   int  shift       // 시프트
   )
```

Parameters

pos

[in] 배열에서 이동된 요소의 위치

shift

[in] 이동 값(양수 및 음수 모두).

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CArrayFloat::Shift(int,int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 시프트
   if(!array.Shift(10,-5))
     {
      printf("시프트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

지정된 배열 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 제거할 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayFloat::Delete(int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.Delete(0))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

지정된 배열 위치에서 요소 그룹을 삭제합니다.

```
bool  DeleteRange(
   int  from,     // 첫 번째 요소의 위치
   int  to        // 최근 요소의 위치
   )
```

Parameters

from

[in] 제거할 첫 번째 배열 요소의 위치.

to

[in] 제거할 마지막 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayFloat::DeleteRange(int,int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.DeleteRange(0,10))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

지정된 배열 위치에서 요소를 가져옵니다.

```
float  At(
   int  pos      // 위치
   ) const
```

Parameters

pos

[in] 배열에서 원하는 요소의 위치.

Return Value

성공시 –요소의 값, 존재하지 않는 위치에서 요소를 가져오려는 시도가 있었을 경우(최근 오류 코드가 ERR_OUT_OF_RANGE) - FLT_MAX

Note

물론 FLT_MAX는 배열 요소의 유효한 값일 수 있습니다. 따라서 이러한 값을 받은 후에는 항상 마지막 에러 코드를 확인해야 합니다.

예제:

```
//--- CArrayFloat::At(int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      float result=array.At(i);
      if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 배열에서 읽는 중 오류 발생
         printf("요소 가져오기 오류");
         delete array;
         return;
        }
      //--- 요소 사용
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const float&  src[]      // 소스 배열
   ) const
```

Parameters

src[]

[in] 비교를 위한 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayFloat::CompareArray(const float &[]) 예제
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArrayconst

배열을 다른 배열과 비교합니다.

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // 소스에 대한 포인터
   ) const
```

Parameters

src

[in] 비교를 위한 요소 소스로 사용되는 CArrayFloat 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayFloat::CompareArray(const CArrayFloat*) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

정렬된 배열에 요소를 삽입합니다.

```
bool  InsertSort(
   float  element      // 삽입할 요소
   )
```

Parameters

element

[in] 정렬된 배열에 삽입할 요소의 값

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayFloat::InsertSort(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 삽입
   if(!array.InsertSort(100.0))
     {
      printf("삽입 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 배열에서 샘플과 같은 요소를 검색합니다.

```
int  Search(
   float  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayFloat::Search(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.Search(100.0)!=-1) printf("요소가 검색됨");
   else                        printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다.

```
int  SearchGreat(
   float  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayFloat::SearchGreat(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreat(100.0)!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다.

```
int  SearchLess(
   float  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayFloat:: SearchLess(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLess(100.0)!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchGreatOrEqual(
   float  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayFloat::SearchGreatOrEqual(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("요소가 검색됨");
   else                                    printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchLessOrEqual(
   float  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayFloat::SearchLessOrEqual(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLessOrEqual(100.0)!=-1) printf("요소가 검색됨");
   else                                   printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다.

```
int  SearchFirst(
   float  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayFloat::SearchFirst(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchFirst(100.0)!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다.

```
int  SearchLast(
   float  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayFloat::SearchLast(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLast(100.0)!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

배열에서 샘플과 동일한 요소를 검색합니다.

```
int  SearchLinear(
   float  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소입니다.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

Note

이 메소드는 정렬되지 않은 배열에 선형 검색(또는 순차 검색) 알고리즘을 사용합니다.

예제:

```
//--- CArrayFloat::SearchLinear(float) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 검색
   if(array.SearchLinear(100.0)!=-1) printf("요소가 검색됨");
   else                              printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayFloat::Save(int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 100 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayFloat::Load(int) 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 요소 사용
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

배열 유형 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

배열 유형 식별자(CArrayFloat -87용).

예제:

```
//--- CArrayFloat::Type() 예제
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 유형 가져오기
   int type=array.Type();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayDouble

CArrayDouble 클래스는 이중 변수의 동적 배열 클래스입니다.

Description

CArrayDouble 클래스는 이중 변수의 동적 배열로 작업할 수 있는 기능을 제공합니다. 클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

Declaration

```
class CArrayDouble : public CArray
```

Title

```
#include <Arrays\ArrayDouble.mqh>
```

상속 계층

CObject
  CArray
    CArrayDouble

직계 종속

CDoubleBuffer

그룹별 클래스 메서드

Attributes	
Delta	비교 허용오차 설정
메모리 컨트롤	
Reserve	메모리를 할당하여 배열 크기를 늘립니다
Resize	배열의 새(작은) 크기를 설정합니다
Shutdown	전체 메모리 릴리스로 배열을 지웁니다
메서드 추가	
Add	배열 끝에 요소를 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
Insert	요소를 배열의 지정된 위치에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
메서드 업데이트	
Update	지정된 배열 위치에서 요소를 변경합니다
Shift	배열의 지정된 위치에서 지정된 오프셋으로 요소를 이동합니다
메서드 삭제	
Delete	지정된 배열 위치에서 요소를 제거합니다
DeleteRange	지정된 배열 위치에서 요소 그룹을 삭제합니다
메서드 액세스	
At	지정된 배열 위치에서 요소를 가져옵니다
메서드 비교	
CompareArray	배열을 다른 배열과 비교합니다
CompareArray	배열을 다른 배열과 비교합니다
최소/최대 찾기	
Minimum	지정된 범위에서 가장 낮은 값 색인을 가져옵니다
Maximum	지정된 범위에서 가장 높은 값 색인을 가져옵니다
정렬된 배열 작업	
InsertSort	정렬된 배열에 요소를 삽입합니다
Search	정렬된 배열에서 샘플과 동일한 요소를 검색합니다
SearchGreat	정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다
SearchLess	정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다
SearchGreatOrEqual	정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다
SearchLessOrEqual	정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다
SearchFirst	정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다
SearchLast	정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다
SearchLinear	배열에서 샘플과 동일한 요소를 검색합니다
Input/output	
virtual Save	데이터 배열을 파일에 저장합니다
virtual Load	파일에서 데이터 배열을 로드합니다
virtual Type	배열의 유형 식별자를 가져옵니다

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Delta

비교 허용오차를 설정합니다.

```
void  Delta(
   double  delta      // 허용오차
   )
```

Parameters

delta

[in] 비교 허용오차의 새 값.

Return Value

No

Note

비교 허용 오차는 검색에 사용됩니다. 값의 차이가 허용오차보다 작거나 같으면 같은 것으로 간주됩니다. 기본 허용오차 값은 0.0입니다.

예제:

```
//--- CArrayDouble::Delta(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 비교 변화량 설정
   array.Delta(0.001);
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

메모리를 할당하여 배열 크기를 늘립니다.

```
bool  Reserve(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 추가 요소 수.

Return Value

성공적이면 true, 0보다 작거나 혹은 같은 양을 요청하려고 했거나 배열을 늘리지 못했을 경우 false.

Note

메모리 조각화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16을 사용하여 배열 크기를 변경합니다.

예제:

```
//--- CArrayDouble::Reserve(int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 메모리 보존
   if(!array.Reserve(1024))
     {
      printf("보존 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

배열의 새(작은) 크기를 설정합니다.

```
bool  Resize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 배열의 새 크기.

Return Value

성공적이면 true, 크기를 0보다 작게 설정하려는 시도가 있을 경우 false.

Note

배열의 크기를 변경하면 메모리를 최적으로 사용할 수 있습니다. 정량보다 과도한 요소는 손실됩니다. 메모리 파편화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16에 따라 배열 크기가 변경됩니다.

예제:

```
//--- CArrayDouble::Resize(int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 리사이즈
   if(!array.Resize(10))
     {
      printf("리사이즈 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

전체 메모리 릴리스로 배열을 지웁니다.

```
bool  Shutdown()
```

Return Value

성공이면 true, 오류면 false.

예제:

```
//--- CArrayDouble::Shutdown() 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 셧다운
   if(!array.Shutdown())
     {
      printf("셧다운 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

배열 끝에 요소를 추가합니다.

```
bool  Add(
   double  element      // 추가할 요소
   )
```

Parameters

element

[in] 배열에 추가할 요소의 값.

Return Value

성공적이면 true, 요소를 추가할 수 없으면 false.

예제:

```
//--- CArrayDouble::Add(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("요소 추가 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const double&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 추가할 소스 요소의 배열에 대한 참조.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayDouble::AddArray(const double &[]) 예제
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const CArrayDouble*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 추가할 요소의 소스로 사용되는 CARayDouble 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayDouble::AddArray(const CArrayDouble*) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete src;
      delete array;
      return;
     }
   //--- 소스 배열 삭제
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

요소를 배열의 지정된 위치에 삽입합니다.

```
bool  Insert(
   double  element,     // 삽입할 요소
   int     pos          // 위치
   )
```

Parameters

element

[in] 배열에 삽입할 요소의 값

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayDouble::Insert(double,int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("삽입 실패");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   const double&  src[],     // 소스 배열
   int            pos        // 포지션
   )
```

Parameters

src[]

[in] 삽입할 요소의 소스로 사용되는 배열에 대한 참조

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayDouble::InsertArray(const double &[],int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   CArrayDouble*  src,     // 소스에 대한 포인터
   int            pos      // 위치
   )
```

Parameters

src

[in] 삽입할 요소의 소스로 사용되는 CArrayDouble 클래스의 인스턴스에 대한 포인터.

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayDouble::InsertArray(const CArrayDouble*,int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const double&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 복사할 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayDouble::AssignArray(const double &[]) 예제
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const CArrayDouble*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 복사할 요소의 소스로 사용되는 CARayDouble 클래스의 인스턴스에 대한 포인터.

Return Value

성공시 true, 요소 복제 실패시 false.

예제:

```
//--- CArrayDouble::AssignArray(const CArrayDouble*) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayDouble *src  =new CArrayDouble;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete src;
      delete array;
      return;
     }
   //--- 어레이가 동일
   //--- 소스 배열 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

지정된 배열 위치에서 요소를 변경합니다.

```
bool  Update(
   int     pos,          // 위치
   double  element       // 값
   )
```

Parameters

pos

[in] 배열에서 변경할 요소의 위치

element

[in] 요소의 새 값.

Return Value

성공시 true, 요소 변경 불가시 false.

예제:

```
//--- CArrayDouble::Update(int,double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 업데이트
   if(!array.Update(0,100.0))
     {
      printf("업데이트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

배열의 지정된 위치에서 지정된 오프셋으로 항목을 이동합니다.

```
bool  Shift(
   int  pos,        // 위치
   int  shift       // 시프트
   )
```

Parameters

pos

[in] 배열에서 이동된 요소의 위치

shift

[in] 이동 값(양수 및 음수 모두).

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CArrayDouble::Shift(int,int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 시프트
   if(!array.Shift(10,-5))
     {
      printf("시프트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

지정된 배열 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 제거할 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayDouble::Delete(int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.Delete(0))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

지정된 배열 위치에서 요소 그룹을 삭제합니다.

```
bool  DeleteRange(
   int  from,     // 첫 번째 요소의 위치
   int  to        // 마지막 요소의 위치
   )
```

Parameters

from

[in] 제거할 첫 번째 배열 요소의 위치.

to

[in] 제거할 마지막 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayDouble::DeleteRange(int,int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.DeleteRange(0,10))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

지정된 배열 위치에서 요소를 가져옵니다.

```
double  At(
   int  pos      // 위치
   ) const
```

Parameters

pos

[in] 배열에서 원하는 요소의 위치.

Return Value

성공시 요소의 값, 존재하지 않는 위치에서 요소를 가져오려는 시도가 있었을 시(최근 오류 코드가 ERR_OUT_OF_RANGE) - DBL_MAX.

Note

물론 DBL_MAX는 배열 요소의 유효한 값일 수 있습니다. 따라서 이러한 값을 받은 후에는 항상 마지막 에러 코드를 확인해야 합니다.

예제:

```
//--- CArrayDouble::At(int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      double result=array.At(i);
      if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  배열에서 읽는 중 오류 발생
         printf("요소 가져오기 오류");
         delete array;
         return;
        }
      //--- 요소 사용
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const double&  src[]      // 소스 배열
   ) const
```

Parameters

src[]

[in] 비교를 위한 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayDouble::CompareArray(const double &[]) 예제
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const CArrayDouble*  src      // 소스에 대한 포인터
   ) const
```

Parameters

src

[in] 비교를 위한 요소 소스로 사용되는 CArrayDouble 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayDouble::CompareArray(const CArrayDouble*) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Minimum

지정된 범위에서 배열의 가장 낮은 요소에 대한 색인을 가져옵니다.

```
int  Minimum(
   int  start,      // 시작 인덱스
   int  count       // 요소 수
   ) const
```

Parameters

start

[in] 검색 범위의 시작 인덱스.

count

[in] 검색 범위 크기(요소 수).

Return Value

지정한 범위에서 가장 낮은 요소의 인덱스.

© 2000-2025, MetaQuotes Ltd.

Maximum

지정된 범위에서 배열의 가장 높은 요소에 대한 색인을 가져옵니다.

```
int  Maximum(
   int  start,      // 시작 인덱스
   int  count       // 요소 수
   ) const
```

Parameters

start

[in] 검색 범위의 시작 인덱스.

count

[in] 검색 범위 크기(요소 수).

Return Value

지정된 범위에서 가장 높은 요소의 색인.

© 2000-2025, MetaQuotes Ltd.

InsertSort

정렬된 배열에 요소를 삽입합니다.

```
bool  InsertSort(
   double  element      // 삽입할 요소
   )
```

Parameters

element

[in] 정렬된 배열에 삽입할 요소의 값

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayDouble::InsertSort(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 삽입
   if(!array.InsertSort(100.0))
     {
      printf("삽입 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 배열에서 샘플과 같은 요소를 검색합니다.

```
int  Search(
   double  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayDouble::Search(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.Search(100.0)!=-1) printf("요소가 검색됨");
   else                        printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다.

```
int  SearchGreat(
   double  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayDouble::SearchGreat(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreat(100.0)!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다.

```
int  SearchLess(
   double  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayDouble::SearchLess(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLess(100.0)!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchGreatOrEqual(
   double  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayDouble::SearchGreatOrEqual(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("요소가 검색됨");
   else                                    printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchLessOrEqual(
   double  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayDouble::SearchLessOrEqual(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLessOrEqual(100.0)!=-1) printf("요소가 검색됨");
   else                                   printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다.

```
int  SearchFirst(
   double  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayDouble::SearchFirst(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchFirst(100.0)!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다.

```
int  SearchLast(
   double  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayDouble::SearchLast(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLast(100.0)!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

배열에서 샘플과 동일한 요소를 검색합니다.

```
int  SearchLinear(
   double  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

Note

이 메소드는 정렬되지 않은 배열에 선형 검색(또는 순차 검색) 알고리즘을 사용합니다.

예제:

```
//--- CArrayDouble::SearchLinear(double) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 검색
   if(array.SearchLinear(100.0)!=-1) printf("요소가 검색됨");
   else                              printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayDouble::Save(int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int            file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 100 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayDouble::Load(int) 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 요소 사용
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

배열 유형 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

배열 유형 식별자(CArrayDouble -87용).

예제:

```
//--- CArrayDouble::Type() 예제
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 유형 가져오기
   int type=array.Type();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayString

CArrayString 클래스는 문자열 변수의 동적 배열 클래스입니다.

Description

CArrayString 클래스는 동적 문자열 변수 배열로 작업할 수 있는 기능을 제공합니다. 클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

Declaration

```
class CArrayString : public CArray
```

Title

```
#include <Arrays\ArrayString.mqh>
```

상속 계층

CObject
  CArray
    CArrayString

그룹별 클래스 메서드

메모리 컨트롤	
Reserve	메모리를 할당하여 배열 크기를 늘립니다
Resize	배열의 새(작은) 크기를 설정합니다
Shutdown	전체 메모리 릴리스로 배열을 지웁니다
메서드 추가	
Add	배열 끝에 요소를 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
AddArray	한 배열의 요소를 다른 배열의 끝에 추가합니다
Insert	요소를 배열의 지정된 위치에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
메서드 업데이트	
Update	지정된 위치 배열에서 요소를 변경합니다

© 2000-2025, MetaQuotes Ltd.

메모리 컨트롤	
Shift	배열의 지정된 위치에서 지정된 오프셋으로 항목을 이동합니다
메서드 삭제	
Delete	지정된 배열 위치에서 요소를 제거합니다
DeleteRange	지정된 배열 위치에서 요소 그룹을 삭제합니다
메서드 액세스	
At	지정된 배열 위치에서 요소를 가져옵니다
메서드 비교	
CompareArray	배열을 다른 배열과 비교합니다
CompareArray	배열을 다른 배열과 비교합니다
정렬된 배열 연산	
InsertSort	정렬된 배열에 요소를 삽입합니다
Search	정렬된 배열에서 샘플과 동일한 요소를 검색합니다
SearchGreat	정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다
SearchLess	정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다
SearchGreatOrEqual	정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다
SearchLessOrEqual	정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다
SearchFirst	정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다
SearchLast	정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다
SearchLinear	배열에서 샘플과 동일한 요소를 검색합니다
Input/output	
virtual Save	데이터 배열을 파일에 저장합니다
virtual Load	파일에서 데이터 배열을 로드합니다
virtual Type	배열의 유형 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

Reserve

메모리를 할당하여 배열 크기를 늘립니다.

```
bool  Reserve(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 추가 요소 수.

Return Value

성공적이면 true, 0보다 작거나 혹은 같은 양을 요청하려고 했거나 배열을 늘리지 못했을 경우 false.

Note

메모리 조각화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16을 사용하여 배열 크기를 변경합니다.

예제:

```
//--- CArrayString::Reserve(int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 메모리 보존
   if(!array.Reserve(1024))
     {
      printf("보존 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

배열의 새(작은) 크기를 설정합니다.

```
bool  Resize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 배열의 새 크기.

Return Value

성공적이면 true, 크기를 0보다 작게 설정하려는 시도가 있을 경우 false.

Note

배열의 크기를 변경하면 메모리를 최적으로 사용할 수 있습니다. 정량보다 과도한 요소는 손실됩니다. 메모리 파편화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16에 따라 배열 크기가 변경됩니다.

예제:

```
//--- CArrayString::Resize(int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 리사이즈
   if(!array.Resize(10))
     {
      printf("리사이즈 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

전체 메모리 릴리스로 배열을 지웁니다.

```
bool  Shutdown()
```

Return Value

성공이면 true, 오류면 false.

예제:

```
//--- CArrayString::Shutdown() 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 셧다운
   if(!array.Shutdown())
     {
      printf("셧다운 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

배열 끝에 요소를 추가합니다.

```
bool  Add(
   string  element      // 추가할 요소
   )
```

Parameters

element

[in] 배열에 추가할 요소의 값.

Return Value

성공적이면 true, 요소를 추가할 수 없으면 false.

예제:

```
//--- CArrayString::Add(string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      if(!array.Add(IntegerToString(i)))
        {
         printf("요소 추가 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const string&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 추가할 소스 요소의 배열에 대한 참조.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayString::AddArray(const string &[]) 예제
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const CArrayString*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 추가할 요소의 소스로 사용되는 CArrayString 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 추가 불가시 false.

예제:

```
//--- CArrayString::AddArray(const CArrayString*) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 추가
   if(!array.AddArray(src))
     {
      printf("배열 추가 실패");
      delete src;
      delete array;
      return;
     }
   //--- 소스 배열 삭제
   delete src;
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

요소를 배열의 지정된 위치에 삽입합니다.

```
bool  Insert(
   string  element,     // 삽입할 요소
   int     pos          // 위치
   )
```

Parameters

element

[in] 배열에 삽입할 요소의 값

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayString::Insert(string,int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(IntegerToString(i),0))
        {
         printf("삽입 실패");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   const string&  src[],     // 소스 배열
   int            pos        // 위치
   )
```

Parameters

src[]

[in] 삽입할 요소의 소스로 사용되는 배열에 대한 참조

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayString::InsertArray(const string &[],int) 예제
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   CArrayString*  src,     // 소스에 대한 포인터
   int             pos      // 위치
   )
```

Parameters

src

[in] 삽입할 요소의 소스로 사용되는 CArrayString 클래스의 인스턴스에 대한 포인터.

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

예제:

```
//--- CArrayString::InsertArray(const CArrayString*,int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열 삽입
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete src;
      delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
   //--- 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const string&  src[]      // 소스 배열
   )
```

Parameters

src[]

[in] 복사할 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayString::AssignArray(const string &[]) 예제
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

한 배열의 요소를 다른 배열로 복사합니다.

```
bool  AssignArray(
   const CArrayString*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 복사할 요소의 소스로 사용되는 CArrayString 클래스의 인스턴스에 대한 포인터.

Return Value

성공시 true, 요소 복제 실패시 false.

예제:

```
//--- CArrayString::AssignArray(const CArrayString*) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayString *src  =new CArrayString;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
      printf("배열 할당 오류");
      delete src;
      delete array;
      return;
     }
   //--- 어레이가 동일
   //--- 소스 배열 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

지정된 배열 위치에서 요소를 변경합니다.

```
bool  Update(
   int     pos,          // 위치
   string  element       // 값
   )
```

Parameters

pos

[in] 배열에서 변경할 요소의 위치

element

[in] 요소의 새 값

Return Value

성공시 true, 요소 변경 불가시 false.

예제:

```
//--- CArrayString::Update(int, string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 업데이트
   if(!array.Update(0,"ABC"))
     {
      printf("업데이트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

배열의 지정된 위치에서 지정된 오프셋으로 항목을 이동합니다.

```
bool  Shift(
   int  pos,        // 위치
   int  shift       // 시프트
   )
```

Parameters

pos

[in] 배열에서 이동된 요소의 위치

shift

[in] 이동 값(양수 및 음수 모두).

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CArrayString::Shift(int,int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 시프트
   if(!array.Shift(10,-5))
     {
      printf("시프트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

지정된 배열 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 제거할 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayString::Delete(int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.Delete(0))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

지정된 배열 위치에서 요소 그룹을 삭제합니다.

```
bool  DeleteRange(
   int  from,     // 첫 번째 요소의 위치
   int  to        // 최근 요소의 위치
   )
```

Parameters

from

[in] 제거할 첫 번째 배열 요소의 위치.

to

[in] 제거할 마지막 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

예제:

```
//--- CArrayString::DeleteRange(int,int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.DeleteRange(0,10))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

지정된 배열 위치에서 요소를 가져옵니다.

```
string  At(
   int  pos      // 위치
   ) const
```

Parameters

pos

[in] 배열에서 원하는 요소의 위치.

Return Value

성공이면 요소의 값, 존재하지 않는 위치에서 요소를 가져오려는 시도가 있었을 경우(마지막 오류 코드가 ERR_OUT_OF_RANGE) - ""

Note

물론, ""는 배열 요소의 유효한 값일 수 있습니다. 따라서 이러한 값을 받은 후에는 항상 마지막 에러 코드를 확인해야 합니다.

예제:

```
//--- CArrayString::At(int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      string result=array.At(i);
      if(result=="" && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- 배열에서 읽는 중 오류 발생
         printf("요소 가져오기 오류");
         delete array;
         return;
        }
      //--- 요소 사용
      //--- . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const string&  src[]      // 소스 배열
   ) const
```

Parameters

src[]

[in] 비교를 위한 요소의 소스로 사용되는 배열에 대한 참조.

Return Value

배열이 같을 시 true, 배열이 같지 않을 시 false.

예제:

```
//--- CArrayString::CompareArray(const string &[]) 예제
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArrays(
   const CArrayString*  src      // 소스에 대한 포인터
   ) const
```

Parameters

src

[in] 비교를 위한 요소 소스로 사용되는 CArrayString 클래스의 인스턴스에 대한 포인터.

Return Value

성공적이면 true, 항목 복제 불가시 false.

예제:

```
//--- CArrayString::CompareArray(const CArrayString*) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 배열 요소 추가
   //--- . . .
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

정렬된 배열에 요소를 삽입합니다.

```
bool  InsertSort(
   string  element      // 삽입할 요소
   )
```

Parameters

element

[in] 정렬된 배열에 삽입할 요소의 값

Return Value

성공시 true, 요소 삽입 실패시 false.

예제:

```
//--- CArrayString::InsertSort(string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 삽입
   if(!array.InsertSort("ABC"))
     {
      printf("삽입 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 배열에서 샘플과 같은 요소를 검색합니다.

```
int  Search(
   string  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayString::Search(string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.Search("ABC")!=-1) printf("요소가 검색됨");
   else                        printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다.

```
int  SearchGreat(
   string  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayString::SearchGreat(string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreat("ABC")!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다.

```
int  SearchLess(
   string  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayString:: SearchLess(string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLess("ABC")!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchGreatOrEqual(
   string  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayString::SearchGreatOrEqual(string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchGreatOrEqual("ABC")!=-1) printf("요소가 검색됨");
   else                                    printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchLessOrEqual(
   string  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayString::  SearchLessOrEqual(string)  예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLessOrEqual("ABC")!=-1) printf("요소가 검색됨");
   else                                   printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다.

```
int  SearchFirst(
   string  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayString::  SearchFirst(string)  예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchFirst("ABC")!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다.

```
int  SearchLast(
   string  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayString:: SearchLast(string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 검색
   if(array.SearchLast("ABC")!=-1) printf("요소가 검색됨");
   else                            printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLinear

배열에서 샘플과 동일한 요소를 검색합니다.

```
int  SearchLinear(
   string  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

Note

이 메소드는 정렬되지 않은 배열에 선형 검색(또는 순차 검색) 알고리즘을 사용합니다.

예제:

```
//--- CArrayString::SearchLinear(string) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 검색
   if(array.SearchLinear("ABC")!=-1) printf("요소가 검색됨");
   else                              printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayString::Save(int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 100 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      array.Add(IntegerToString(i));
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayString::Load(int) 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int            file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 요소 사용
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%s'",i,array.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

배열 유형 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

배열 유형 식별자(CArrayString -89용).

예제:

```
//--- CArrayString::Type() 예제
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 유형 가져오기
   int type=array.Type();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CArrayObj

CArrayObj 클래스는 CObject 및 파생 클래스의 인스턴스에 대한 포인터의 동적 배열 클래스입니다.

Description

클래스 CArrayObj는 CObject 및 그 파생 클래스의 인스턴스에 대한 포인터의 동적 배열로 작업할 수 있는 기능을 제공합니다. 이를 통해 원시 데이터 유형의 다차원 동적 배열과 데이터 구성이 더 복잡한 데이터 구조를 모두 사용할 수 있습니다.

클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

CArrayObj 클래스에는 특정한 세부 요소이 있습니다.한

Declaration

```
class CArrayObj : public CArray
```

Title

```
#include <Arrays\ArrayObj.mqh>
```

상속 계층

CObject
  CArray
    CArrayObj

직계 종속

CIndicators, CSeries

그룹별 클래스 메서드

Attributes	
FreeMode	메모리 관리 플래그를 가져옵니다
FreeMode	메모리 관리 플래그를 설정합니다
메모리 컨트롤	
Reserve	메모리를 할당하여 배열 크기를 늘립니다
Resize	배열의 새(작은) 크기를 설정합니다
Shutdown	메모리(요소 제외)의 전체 할당 해제로 배열을 지웁니다.
새 요소 생성	
virtual CreateElement	지정된 위치에 새 배열 요소를 작성합니다
메서드 추가	

© 2000-2025, MetaQuotes Ltd.

Attributes	
Add	배열 끝에 요소를 추가합니다
AddArray	배열 끝에 요소를 추가합니다
Insert	요소를 배열의 지정된 위치에 삽입합니다
InsertArray	지정된 위치에서 다른 배열의 배열 요소에 삽입합니다
AssignArray	한 배열의 요소를 다른 배열로 복사합니다
메서드 업데이트	
Update	지정된 위치 배열에서 요소를 변경합니다
Shift	배열의 지정된 위치에서 지정된 오프셋으로 항목을 이동합니다
메서드 삭제	
Detach	지정된 위치에서 요소를 가져와 배열에서 제거합니다
Delete	지정된 배열 위치에서 요소를 제거합니다
DeleteRange	지정된 배열 위치에서 요소 그룹을 삭제합니다
Clear	배열 메모리를 해제하지 않고 배열의 모든 요소를 제거합니다
메서드 액세스	
At	지정된 배열 위치에서 요소를 가져옵니다
메서드 비교	
CompareArray	배열을 다른 배열과 비교합니다
정렬된 배열 작업	
InsertSort	정렬된 배열에 요소를 삽입합니다
Search	정렬된 배열에서 샘플과 동일한 요소를 검색합니다
SearchGreat	정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다
SearchLess	정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다
SearchGreatOrEqual	정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다
SearchLessOrEqual	정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
SearchFirst	정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다
SearchLast	정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다
Input/output	
Save	데이터 배열을 파일에 저장합니다
Load	파일에서 데이터 배열을 로드합니다
Type	배열의 유형 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CObject 클래스의 배열은 실용적인 응용 프로그램(표준 라이브러리의 모든 클래스 포함)을 가지고 있습니다.

예를 들어, 2차원 배열에 대한 옵션을 생각해 보십시오:

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int i,j;
   int first_size=10;
   int second_size=100;
//--- 배열 생성
   CArrayObj    *array=new CArrayObj;
   CArrayDouble *sub_array;
//---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
//--- 하위 배열 생성
   for(i=0;i<first_size;i++)
     {
      sub_array=new CArrayDouble;
      if(sub_array==NULL)
        {
         delete array;
         printf("개체 생성 오류");
```

© 2000-2025, MetaQuotes Ltd.

```
         return;
        }
      //--- 배열 채우기
      for(j=0;j<second_size;j++)
        {
         sub_array.Add(i*j);
        }
      array.Add(sub_array);
     }
//--- 배열 생성 OK
   for(i=0;i<first_size;i++)
     {
      sub_array=array.At(i);
      for(j=0;j<second_size;j++)
        {
         double element=sub_array.At(j);
         //--- 배열 요소 사용
        }
     }
   delete array;
  }
```

Subtleties

클래스에는 동적 메모리를 제어하는 메커니즘이 있으므로 배열 요소를 사용할 때 주의해야 합니다.

메모리 관리 메커니즘은 FreeMode (bool) 방법을 사용하여 켜거나 끌 수 있습니다. 기본적으로 메커니즘은 실행됩니다.

따라서 CArrayObj 클래스를 처리하는 두 가지 옵션이 있습니다:

1. 메모리 관리 메커니즘이 사용되도록 설정되었습니다. (default)

이 경우 CArrayObj는 배열에서 소자를 제거한 후 소자에 사용된 메모리를 해제해야 합니다. 사용자 지정 프로그램은 배열 요소를 해제하지 않아야 합니다.

예제:

```
   int i;
//--- 배열을 생성
   CArrayObj *array=new CArrayObj;
//--- 배열 요소를 채움
   for(i=0;i<10;i++) array.Add(new CObject);
//--- 뭐든 하기
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- 요소로 작업 수행
      . . .
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 배열을 요소와 함께 제거
   delete array;
```

2. 메모리 관리 메커니즘이 비활성화됩니다.

이 경우, CArrayObj는 배열에서 요소 메모리를 제거한 후 할당 해제를 담당하지 않습니다. 또한 사용자 프로그램은 배열 요소의 할당을 취소해야 합니다.

예제:

```
   int i;
//--- 배열을 생성
   CArrayObj *array=new CArrayObj;
//--- 메모리 관리 메커니즘을 비활성화
   array.FreeMode(false);
//--- 배열에 요소를 채움
   for(i=0;i<10;i++) array.Add(new CObject);
//--- 뭐든 하기
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- 요소로 작업 수행
      . . .
     }
//--- 배열 요소 제거
   while(array.Total()) delete array.Detach();
//--- 빈 배열 제거
   delete array;
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

메모리 관리 플래그를 가져옵니다.

```
bool  FreeMode() const
```

Return Value

메모리 관리 플래그.

예제:

```
//--- CArrayObj::FreeMode() 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 자유 모드 플래그를 얻다
   bool array_free_mode=array.FreeMode();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

메모리 관리의 플래그를 설정합니다.

```
void  FreeMode(
   bool  mode      // 새 플래그
   )
```

Parameters

mode

[in] 메모리 관리 플래그의 새 값.

Return Value

None.

Note

메모리 관리 플래그 설정은 CArrayObj 클래스 사용에 중요한 부분입니다. 배열 요소는 동적 객체에 대한 포인터이므로 배열에서 제거할 때 동적 객체로 수행할 작업을 결정하는 것이 중요합니다.

배열에서 요소를 제거하는 플래그가 설정되어 있으면 삭제 연산자에 의해 요소가 자동으로 삭제됩니다. 플래그가 설정되지 않은 경우 삭제된 객체에 대한 포인터는 여전히 사용자 프로그램의 어딘가에 있으며 나중에 프로그램에 의해 할당 해제된다고 가정합니다.

사용자 프로그램이 메모리 관리 플래그를 재설정할 경우 사용자는 프로그램 종료 전에 어레이 제거에 대한 자신의 책임을 이해해야 합니다. 그렇지 않으면 새 연산자가 요소에 할당한 메모리가 해제되지 않습니다.

대용량 데이터의 경우 터미널이 손상될 수도 있습니다.

사용자가 메모리 관리 플래그를 재설정하지 않으면 다른 함정이 있는 것입니다. 배열의 포인터 요소가 로컬 변수의 어딘가에 저장되면 배열을 제거하면 사용자 프로그램의 심각한 오류와 충돌이 발생합니다. 기본적으로 메모리 관리 플래그가 설정됩니다. 즉, 배열 클래스는 메모리 요소를 비우는 역할을 합니다.

예제:

```
//--- CArrayObj::FreeMode(bool) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 자유 모드 플래그 재설정
   array.FreeMode(false);
   //--- 배열 사용
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Reserve

메모리를 할당하여 배열 크기를 늘립니다.

```
bool  Reserve(
   int  size      // 숫자
   )
```

Parameters

size

[in] 배열의 추가 요소 수.

Return Value

성공적이면 true, 0보다 작거나 혹은 같은 양을 요청하려고 했거나 배열을 늘리지 못했을 경우 false.

Note

메모리 조각화를 줄이기 위해 Step(int) 메서드로 이전에 결정된 단계 또는 기본 단계인 16을 사용하여 배열 크기를 변경합니다.

예제:

```
//--- CArrayObj::Reserve(int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   if(!array.Reserve(1024))
     {
      printf("보존 오류");
      delete array;
      return;
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Resize

배열의 새(작은) 크기를 설정합니다.

```
bool  Resize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 배열의 새 크기.

Return Value

성공적이면 true, 크기를 0보다 작게 설정하려는 시도가 있을 경우 false.

Note

배열의 크기를 변경하면 메모리를 최적으로 사용할 수 있습니다. 정량보다 과도한 요소는 손실됩니다. 메모리 관리 모드에 따라 손실된 요소의 메모리가 해제되거나 해제되지 않습니다.

메모리 파편화를 줄이기 위해 배열의 크기를 변경하려면 이전에 단계(int) 또는 16(기본값) 방법을 통해 주어진 단계를 사용하여 변경하십시오.

예제:

```
//--- CArrayObj::Resize(int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 리사이즈
   if(!array.Resize(10))
     {
      printf("리사이즈 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

메모리 배열을 해제하지 않고 배열의 모든 요소를 제거합니다.

```
void  Clear()
```

Return Value

No.

Note

메모리 관리 플래그를 사용하도록 설정하면 삭제된 개체에 사용된 메모리가 해제됩니다.

예제:

```
//--- CArrayObj::Clear() 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정리
   array.Clear();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shutdown

어레이에 대한 메모리 할당 해제(요소 제외)를 통해 어레이를 지웁니다.

```
bool  Shutdown()
```

Return Value

성공적이면 true, 오류 발생시 false.

Note

메모리 관리를 사용하도록 설정하면 삭제된 요소의 메모리가 할당 취소됩니다.

예제:

```
//--- CArrayObj::Shutdown() 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 셧다운
   if(!array.Shutdown())
     {
      printf("셧다운 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateElement

지정한 위치에 새 배열 요소를 작성합니다.

```
bool  CreateElement(
   int  index      // 위치
   )
```

Parameters

index

[in] 새 요소를 작성할 위치.

Return Value

성공적이면 true, 요소를 생성할 수 없으면 false.

Note

CArrayObj 클래스의 메서드 CreateElement(int)는 항상 false를 반환하며 어떠한 작업도 수행하지 않습니다. 필요한 경우 CreateElement(int) 메서드를 파생 클래스에서 구현해야 합니다.

예제:

```
//--- CArrayObj::CreateElement(int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         size=100;
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 채우기
   array.Reserve(size);
   for(int i=0;i<size;i++)
     {
      if(!array.CreateElement(i))
        {
         printf("요소 생성 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
```

© 2000-2025, MetaQuotes Ltd.

```
   }
```

© 2000-2025, MetaQuotes Ltd.

Add

배열 끝에 요소를 추가합니다.

```
bool  Add(
   CObject*  element      // 추가할 요소
   )
```

Parameters

element

[in] 배열에 추가할 요소의 값.

Return Value

성공적이면 true, 요소를 추가할 수 없으면 false.

Note

잘못된 포인터(예: NULL)가 매개 변수로 전달되면 요소가 배열에 추가되지 않습니다.

예제:

```
//--- CArrayObj::Add(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 100 배열 요소 추가
   for(int i=0;i<100;i++)
     {
      if(!array.Add(new CObject))
        {
         printf("요소 추가 오류");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AddArray

한 배열의 요소를 다른 배열의 끝에 추가합니다.

```
bool  AddArray(
   const CArrayObj *  src      // 소스에 대한 포인터 배열
   )
```

Parameters

src

[in] CArrayDouble 클래스 인스턴스에 대한 포인터 -추가할 요소의 출처.

Return Value

성공적이면 true, 항목 추가 불가시 false.

Note

요소를 배열에서 배열로 추가하는 것은 실제로 포인터를 복사하는 것입니다. 그러므로 메소드를 호출할 때 함정이 존재하며 둘 이상의 변수에 동적 객체에 대한 포인터가 있을 수 있습니다.

```
//--- 예제
extern bool       make_error;
extern int        error;
extern CArrayObj *src;
//--- 새 인스턴스 CArrayObj 를 생성
//--- 기본 메모리 관리가 켜져 있습니다
CArrayObj *array=new CArrayObj;
//--- 소스 배열에서 요소 추가(복사)
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- 잘못된 동작
   switch(error)
     {
      case 0:
         //--- 메모리 관리 플래그를 확인하지 않고 소스 어레이 제거
         delete src;
         //--- 결과:
         //--- 수신기 배열에서 잘못된 포인터로 요소의 주소를 지정할 수 있습니다
         break;
      case 1:
         //--- 소스 어레이에서 메모리 관리 메커니즘 사용 안 함
         if(src.FreeMode()) src.FreeMode(false);
         //--- 그러나 원본 배열은 제거하지 마십시오
         //--- 결과:
         //--- 수신기 배열을 제거한 후 소스 배열에서 잘못된 포인터로 요소의 주소를 지정할 수 있
         break;
      case 2:
```

© 2000-2025, MetaQuotes Ltd.

```
         //--- 소스 어레이에서 메모리 관리 메커니즘 사용 안 함
         src.FreeMode(false);
         //--- 수신기 배열에서 메모리 관리 메커니즘을 사용하지 않도록 설정합니다
         array.FreeMode(false);
         //--- 결과:
         //--- 프로그램 종료 후 "메모리 누수" 발생
         break;
      }
   }
else
   {
    //--- 소스 어레이에서 메모리 관리 메커니즘 사용 안 함
    if(src.FreeMode()) src.FreeMode(false);
    //--- 소스 배열 삭제
    delete src;
    //--- 결과:
    //--- 수신기 배열 요소의 주소가 올바릅니다
    //--- 수신기 배열을 삭제하면 해당 요소가 삭제됩니다
   }
```

예제:

```
//--- CArrayObj::AddArray(const CArrayObj*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
   {
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
      {
       printf("객체 생성 오류");
       return;
      }
    //--- 소스 배열 생성
    CArrayObj *src=new CArrayObj;
    if(src==NULL)
      {
       printf("객체 생성 오류");
       delete array;
       return;
      }
    //--- 자유 모드 플래그 재설정
    src.FreeMode(false);
    //--- 소스 어레이 채우기
    //--- . . .
    //--- 다른 배열 추가
    if(!array.AddArray(src))
```

© 2000-2025, MetaQuotes Ltd.

```
     {
      printf("배열 추가 실패");
      delete src;
      delete array;
      return;
     }
   //--- 요소 없이 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

요소를 배열의 지정된 위치에 삽입합니다.

```
bool  Insert(
   CObject*  element,     // 삽입할 요소
   int       pos          // 위치
   )
```

Parameters

element

[in] 배열에 삽입할 요소의 값

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 요소 삽입 실패시 false.

Note

잘못된 포인터(예: NULL)가 매개 변수로 전달되면 요소가 배열에 추가되지 않습니다.

예제:

```
//--- CArrayObj::Insert(CObject*,int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(new CObject,0))
        {
         printf("삽입 실패");
         delete array;
         return;
        }
     }
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertArray

한 배열의 요소를 다른 배열의 지정된 위치에서 삽입합니다.

```
bool  InsertArray(
   const CArrayObj*  src,     // 소스에 대한 포인터
   int               pos      // 위치
   )
```

Parameters

src

[in] 삽입할 요소의 소스로 사용되는 CArrayObj 클래스의 인스턴스에 대한 포인터.

pos

[in] 삽입할 배열의 위치

Return Value

성공시 true, 항목 삽입 실패시 false.

Note

See: CArrayObj::AddArray(const CArrayObj*).

예제:

```
//--- CArrayObj::InsertArray(const CArrayObj*,int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 자유 모드 플래그 재설정
   src.FreeMode(false);
   //--- 소스 어레이 채우기
   //--- . . .
   //--- 다른 배열 삽입
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!array.InsertArray(src,0))
     {
      printf("배열 삽입 실패");
      delete src;
      delete array;
      return;
     }
   //--- 요소 없이 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

AssignArray

배열을 다른 배열과 비교합니다.

```
bool  AssignArray(
   const CArrayObj*  src      // 소스에 대한 포인터
   )
```

Parameters

src

[in] 복사할 요소의 소스로 사용되는 CArrayObj 클래스의 인스턴스에 대한 포인터.

Return Value

성공시 true, 요소 복제 실패시 false.

Note

AssignArray를 호출할 때 수신기 배열이 비어 있지 않으면 해당 요소가 모두 제거되고 메모리 관리 플래그가 설정되어 있으면 삭제된 요소에 사용된 메모리가 해제됩니다. 수신기 배열은 소스 배열의 정확한 복사본이 됩니다. 또한 CArrayObj::AddArray(const CArrayObj*)을 참조하십시오.

예제:

```
//--- CArrayObj::AssignArray(const CArrayObj*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 자유 모드 플래그 재설정
   src.FreeMode(false);
   //--- 소스 어레이 채우기
   //--- . . .
   //--- 다른 배열을 할당
   if(!array.AssignArray(src))
     {
```

© 2000-2025, MetaQuotes Ltd.

```
      printf("배열 할당 오류");
      delete src;
      delete array;
      return;
     }
   //--- 어레이가 동일
   //--- 요소 없이 소스 배열 삭제
   delete src;
   //--- 배열 사용
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

지정된 배열 위치에서 요소를 변경합니다.

```
bool  Update(
   int       pos,          // 위치
   CObject*  element       // 값
   )
```

Parameters

pos

[in] 배열에서 변경할 요소의 위치

element

[in] 요소의 새 값

Return Value

성공시 true, 요소 변경 불가시 false.

Note

잘못된 포인터(예: NULL)가 매개 변수로 전달되더라도 요소는 변경되지 않습니다. 메모리 관리를 사용하도록 설정하면 교체된 요소의 메모리가 할당 취소됩니다.

예제:

```
//--- CArrayObj::Update(int,CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 업데이트
   if(!array.Update(0,new CObject))
     {
      printf("업데이트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Shift

배열의 지정된 위치에서 지정된 오프셋으로 항목을 이동합니다.

```
bool  Shift(
   int  pos,        // 위치
   int  shift       // 시프트
   )
```

Parameters

pos

[in] 배열에서 이동된 요소의 위치

shift

[in] 이동 값(양수 및 음수 모두).

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CArrayObj::Shift(int,int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 시프트
   if(!array.Shift(10,-5))
     {
      printf("시프트 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Detach

배열의 지정된 위치에서 요소를 제거합니다.

```
CObject*  Detach(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 배열에서 제거된 항목의 위치.

Return Value

성공적이면 제거된 요소에 대한 포인터, 요소 제거 불가시 Null.

Note

요소가 배열에서 제거되면 메모리 관리 플래그에 관계없이 삭제되지 않습니다. 배열 요소 포인터가 사용되면 할당을 취소해야 합니다.

예제:

```
//--- CArrayObj::Detach(int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   CObject *object=array.Detach(0);
   if(object==NULL)
     {
      printf("분리 오류");
      delete array;
      return;
     }
   //--- 요소 사용
   //--- . . .
   //--- 요소 삭제
   delete object;
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

지정된 배열 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 제거할 배열 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

Note

메모리 관리를 사용하도록 설정하면 삭제된 요소의 메모리가 할당 취소됩니다.

예제:

```
//--- CArrayObj::Delete(int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   if(!array.Delete(0))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteRange

지정된 배열 위치에서 요소 그룹을 삭제합니다.

```
bool  DeleteRange(
   int  from,     // 첫 번째 요소의 위치
   int  to        // 최근 요소의 위치
   )
```

Parameters

from

[in] 제거할 첫 번째 배열 요소의 위치.

to

[in] 제거할 마지막 배열 요소의 위치.

Return Value

성공적이면 true, 요소 제거 불가시 false.

Note

메모리 관리를 사용하도록 설정하면 삭제된 요소의 메모리가 할당 취소됩니다.

예제:

```
//--- CArrayObj::DeleteRange(int,int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 요소 삭제
   if(!array.DeleteRange(0,10))
     {
      printf("삭제 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

At

지정된 배열 위치에서 요소를 가져옵니다.

```
CObject*  At(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 배열에서 원하는 요소의 위치.

Return Value

요소의 값(성공적, NULL)이 존재하지 않는 위치의 요소를 가져오려고 했습니다.

예제:

```
//--- CArrayObj::At(int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 요소 추가
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      CObject *result=array.At(i);
      if(result==NULL)
        {
         //--- 배열에서 읽는 중 오류 발생
         printf("요소 가져오기 오류");
         delete array;
         return;
        }
      //--- 요소 사용
      //--- . . .
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareArray

배열을 다른 배열과 비교합니다.

```
bool  CompareArray(
   const CArrayObj*  src      // 소스에 대한 포인터
   ) const
```

Parameters

src

[in] 비교를 위한 요소 소스로 사용되는 CArrayObj 클래스의 인스턴스에 대한 포인터.

Return Value

배열이 같으면 true, 같지 않으면 –.

예제:

```
//--- CArrayObj::CompareArray(const CArrayObj*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 소스 배열 생성
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("객체 생성 오류");
      delete array;
      return;
     }
   //--- 소스 어레이 채우기
   //--- . . .
   //--- 다른 배열과 비교
   int result=array.CompareArray(src);
   //--- 배열 삭제
   delete src;
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

InsertSort

정렬된 배열에 요소를 삽입합니다.

```
bool  InsertSort(
   CObject*  element      // 삽입할 요소
   )
```

Parameters

element

[in] 정렬된 배열에 삽입할 요소의 값

Return Value

성공시 true, 요소 삽입 실패시 false.

Note

잘못된 포인터(예: NULL)가 매개 변수로 전달되면 요소가 배열에 추가되지 않습니다.

예제:

```
//--- CArrayObj::InsertSort(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 요소 삽입
   if(!array.InsertSort(new CObject))
     {
      printf("삽입 오류");
      delete array;
      return;
     }
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 배열에서 샘플과 같은 요소를 검색합니다.

```
int  Search(
   CObject*  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayObj::Search(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 샘플 생성
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("샘플 생성 오류");
      delete array;
      return;
     }
   //--- 표본 속성 설정
   //--- . . .
   //--- 요소 검색
   if(array.Search(sample)!=-1) printf("요소가 검색됨");
   else                         printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreat

정렬된 배열에서 샘플 값을 초과하는 값을 가진 요소를 검색합니다.

```
int  SearchGreat(
   CObject*  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayObj::SearchGreat(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 샘플 생성
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("샘플 생성 오류");
      delete array;
      return;
     }
   //--- 표본 속성 설정
   //--- . . .
   //--- 요소 검색
   if(array.SearchGreat(sample)!=-1) printf("요소가 검색됨");
   else                              printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLess

정렬된 배열에서 샘플 값보다 작은 값을 가진 요소를 검색합니다.

```
int  SearchLess(
   CObject*  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayObj::SearchLess(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 샘플 생성
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("샘플 생성 오류");
      delete array;
      return;
     }
   //--- 표본 속성 설정
   //--- . . .
   //--- 요소 검색
   if(array.SearchLess(sample)!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchGreatOrEqual

정렬된 배열에서 샘플 값보다 크거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchGreatOrEqual(
   CObject*  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayObj::SearchGreatOrEqual(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 샘플 생성
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("샘플 생성 오류");
      delete array;
      return;
     }
   //--- 표본 속성 설정
   //--- . . .
   //--- 요소 검색
   if(array.SearchGreatOrEqual(sample)!=-1) printf("요소가 검색됨");
   else                                     printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLessOrEqual

정렬된 배열에서 샘플 값보다 작거나 같은 값을 가진 요소를 검색합니다.

```
int  SearchLessOrEqual(
   CObject*  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayObj::SearchLessOrEqual(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 샘플 생성
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("샘플 생성 오류");
      delete array;
      return;
     }
   //--- 표본 속성 설정
   //--- . . .
   //--- 요소 검색
   if(array.SearchLessOrEqual(sample)!=-1) printf("요소가 검색됨");
   else                                    printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchFirst

정렬된 배열에서 샘플과 동일한 첫 번째 요소를 검색합니다.

```
int  SearchFirst(
   CObject*  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 -성공, -1 -요소를 찾을 수 없음.

예제:

```
//--- CArrayObj::SearchFirst(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 샘플 생성
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("샘플 생성 오류");
      delete array;
      return;
     }
   //--- 표본 속성 설정
   //--- . . .
   //--- 요소 검색
   if(array.SearchFirst(sample)!=-1) printf("요소가 검색됨");
   else                              printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

SearchLast

정렬된 배열에서 샘플과 동일한 마지막 요소를 검색합니다.

```
int  SearchLast(
   CObject*  element      // 샘플
   ) const
```

Parameters

element

[in] 배열에서 검색할 샘플 요소.

Return Value

찾은 요소의 위치 - 성공, -1 - 요소를 찾을 수 없음.

예제:

```
//--- CArrayObj::SearchLast(CObject*) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 배열 정렬
   array.Sort();
   //--- 샘플 생성
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("샘플 생성 오류");
      delete array;
      return;
     }
   //--- 표본 속성 설정
   //--- . . .
   //--- 요소 검색
   if(array.SearchLast(sample)!=-1) printf("요소가 검색됨");
   else                             printf("요소가 검색되지 않음");
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

데이터 배열을 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

예제:

```
//--- CArrayObj::Save(int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 요소 추가
   //--- . . .
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 데이터 배열을 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료되면 true, 오류면 false.

Note

파일에서 배열 요소를 읽을 때 CArrayObj::CreateElement(int) 메서드를 호출하여 각 요소를 만듭니다.

예제:

```
//--- CArrayObj::Load(int) 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 배열 요소 사용
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- . . .
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

배열 유형 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

배열 유형 식별자(CArrayObj -7778용).

예제:

```
//--- CArrayObj::Type() 예제
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 배열 유형 가져오기
   int type=array.Type();
   //--- 배열 삭제
   delete array;
  }
```

© 2000-2025, MetaQuotes Ltd.

CList

CList 클래스는 CObject 클래스 및 파생 클래스의 동적 인스턴스 목록입니다.

Description

클래스 CList는 CObject 및 파생 클래스의 인스턴스 목록을 작업할 수 있는 기능을 제공합니다. 클래스를 사용하면 배열 요소를 추가/삽입/삭제하고, 배열 정렬을 수행하고, 정렬된 배열에서 검색할 수 있습니다. 또한 파일로 작업하는 방법도 구현되었습니다.

CList 클래스로 작업할 때 몇 가지 세부 요소가 있습니다. 클래스에는 동적 메모리를 제어하는 메커니즘이 있으므로 목록의 요소를 다룰 때 주의해야 합니다.

CArrayObj에 설명된 것과 유사한 메모리 관리 메커니즘의 세부 요소.

Declaration

```
class CList : public CObject
```

Title

```
#include <Arrays\List.mqh>
```

상속 계층

CObject
  CList

그룹별 클래스 메서드

Attributes	
FreeMode	목록 요소를 삭제할 때 메모리 관리 플래그를 가져옵니다.
FreeMode	목록 요소를 삭제할 때 메모리 관리 플래그를 설정합니다.
Total	리스트의 요소 수를 가져옵니다
IsSorted	정렬된 목록 플래그를 가져옵니다
SortMode	정렬 모드를 가져옵니다
메서드 생성	
CreateElement	새 목록 요소를 생성합니다
메서드 추가	
Add	목록의 끝에 요소를 추가합니다
Insert	목록의 지정된 위치에 요소를 삽입합니다
메서드 삭제	
DetachCurrent	요소를 "실제로" 삭제하지 않고 목록의 현재 위치에서 요소를 제거합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
DeleteCurrent	리스트의 현재 위치에서 요소를 제거합니다
Delete	리스트의 지정된 위치에서 요소를 제거합니다
Clear	모든 목록 요소를 제거합니다
Navigation	
IndexOf	지정된 리스트 요소의 색인을 가져옵니다
GetNodeAtIndex	지정된 리스트 요소의 색인을 가져옵니다
GetFirstNode	리스트의 첫 번째 요소를 가져옵니다
GetPrevNode	리스트의 이전 요소를 가져옵니다
GetCurrentNode	현재 리스트 요소를 가져옵니다
GetNextNode	현재 리스트 요소를 가져옵니다
GetLastNode	리스트의 마지막 요소를 가져옵니다
순번 메서드	
Sort	목록 정렬
MoveToIndex	목록의 현재 요소를 지정된 위치로 이동합니다
Exchange	리스트의 두 요소를 스왑합니다
메서드 비교	
CompareList	목록을 다른 목록과 비교합니다
메서드 검색	
Search	정렬된 리스트에서 샘플과 동일한 요소를 검색합니다
Input/output	
virtual Save	목록 데이터를 파일에 저장합니다
virtual Load	파일에서 목록 데이터를 로드합니다
virtual Type	리스트 유형 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

FreeMode

목록 요소를 삭제할 때 메모리 관리 플래그를 가져옵니다.

```
bool  FreeMode() const
```

Return Value

메모리 관리 플래그.

예제:

```
//--- CList::FreeMode() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 자유 모드 플래그 획득
   bool list_free_mode=list.FreeMode();
   //--- 목록 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

FreeMode

목록 요소를 삭제할 때 메모리 관리 플래그를 설정합니다.

```
void  FreeMode(
   bool  mode      // 새 값
   )
```

Parameters

mode

[in] 메모리 관리 플래그의 새 값.

Note

메모리 관리 플래그 설정은 CList 클래스 사용에 있어 중요한 부분입니다. 목록 요소는 동적 객체에 대한 포인터이므로 목록에서 제거할 때 동적 객체로 수행할 작업을 결정하는 것이 중요합니다.

플래그가 설정된 경우 리스트에서 요소를 제거할 때 삭제 연산자에 의해 요소가 자동으로 삭제됩니다. 플래그가 설정되지 않은 경우 삭제된 객체에 대한 포인터는 여전히 사용자 프로그램의 어딘가에 있으며 나중에 프로그램에 의해 할당 해제된다고 가정합니다.

사용자 프로그램이 메모리 관리 플래그를 재설정하는 경우 사용자는 프로그램 종료 전에 목록 요소를 제거해야 할 책임이 있음을 이해해야 합니다. 그렇지 않으면 새 연산자가 요소에 할당한 메모리가 해제되지 않습니다.

많은 양의 데이터의 경우 이는 터미널 충돌로 이어질 수도 있습니다.

사용자 프로그램이 메모리 관리 플래그를 재설정하지 않으면 다른 "위험"이 있는 것입니다. 목록의 포인터 요소가 로컬 변수의 어딘가에 저장되면 목록을 제거하면 사용자 프로그램의 심각한 오류와 충돌이 발생합니다. 기본적으로 메모리 관리 플래그가 설정됩니다. 즉, 목록의 클래스가 메모리 요소를 해제합니다.

예제:

```
//--- CList::FreeMode(bool) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 자유 모드 플래그 재설정
   list.FreeMode(false);
   //--- 목록 사용
   //--- . . .
   //--- 목록 삭제
   delete list;
```

© 2000-2025, MetaQuotes Ltd.

```
}
```

© 2000-2025, MetaQuotes Ltd.

Total

리스트의 요소 수를 가져옵니다.

```
int  Total() const
```

Return Value

목록에 있는 요소 수.

예제:

```
//--- CList::Total() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 총 수 확인
   int total=list.Total();
   //--- 리스트 사용
   //--- ...
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

IsSorted

정렬된 리스트 플래그를 가져옵니다.

```
bool  IsSorted(
   int  mode=0      // 정렬 모드
   ) const
```

Parameters

mode=0

[in] 선택한 정렬 모드.

Return Value

정렬된 목록의 플래그 지정된 모드를 사용하여 목록이 정렬되면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

Note

정렬된 목록의 플래그를 직접 변경할 수 없습니다. 이 플래그는 Sort(int)에 의해 설정되며 추가/삽입 방법으로 재설정됩니다.

예제:

```
//--- CList::IsSorted() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 정렬 점검
   if(list.IsSorted(0))
     {
      //--- 정렬된 목록에 메소드 사용
      //--- ...
     }
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

SortMode

정렬 버전을 가져옵니다.

```
int  SortMode() const
```

Return Value

정렬 모드, 또는 목록이 정렬되지 않은 경우 -1.

예제:

```
//--- CList::SortMode() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 정렬 모드 확인
   int sort_mode=list.SortMode();
   //--- 리스트 사용
   //--- ...
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

CreateElement

목록의 새 요소를 생성합니다.

```
CObject*  CreateElement()
```

Return Value

새로 만든 요소(성공, NULL)에 대한 포인터는 요소를 만들 수 없습니다.

Note

CList 클래스의 메서드 CreateElement()는 항상 NULL을 반환하고 어떠한 작업도 수행하지 않습니다. 필요한 경우 CreateElement() 메서드를 파생 클래스에서 구현해야 합니다.

예제:

```
//--- CList::CreateElement(int) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    size=100;
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 목록 채우기
   for(int i=0;i<size;i++)
     {
      CObject *object=list.CreateElement();
      if(object==NULL)
        {
         printf("요소 생성 오류");
         delete list;
         return;
        }
      list.Add(object);
     }
   //--- 리스트 사용
   //--- . . .
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Add

리스트 끝에 요소를 추가합니다.

```
int  Add(
   CObject*  element      // 추가할 요소
   )
```

Parameters

element

[in] 목록에 추가할 요소의 값.

Return Value

추가된 요소의 인덱스 - 성공, -1 - 오류.

Note

잘못된 포인터(예: NULL)가 매개 변수로 전달되면 요소가 리스트에 추가되지 않습니다.

예제:

```
//--- CList::Add(CObject*) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 100개 요소 추가
   for(int i=0;i<100;i++)
     {
      if(list.Add(new CObject)==-1)
        {
         printf("요소 추가 오류");
         delete list;
         return;
        }
     }
   //--- 리스트 사용
   //--- . . .
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Insert

리스트의 지정된 위치에 요소를 삽입합니다.

```
int  Insert(
   CObject*  element,     // 삽입할 요소
   int       pos          // 위치
   )
```

Parameters

element

[in] 목록에 삽입할 요소의 값

pos

[in] 삽입할 목록의 위치

Return Value

삽입된 요소의 색인 -성공, -1 -오류.

Note

잘못된 포인터(예: NULL)가 매개 변수로 전달되면 요소가 리스트에 추가되지 않습니다.

예제:

```
//--- CList::Insert(CObject*,int) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 100개 요소 삽입
   for(int i=0;i<100;i++)
     {
      if(list.Insert(new CObject,0)==-1)
        {
         printf("요소 삽입 오류");
         delete list;
         return;
        }
     }
   //--- 리스트 사용
   //--- . . .
   //--- 리스트 삭제
```

© 2000-2025, MetaQuotes Ltd.

```
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

DetachCurrent

목록의 현재 위치에서 "실제" 삭제 없이 요소를 추출합니다.

```
CObject*  DetachCurrent()
```

Return Value

성공한 경우 제거된 요소에 대한 포인터, NULL -요소를 제거할 수 없음.

Note

목록에서 제거해도 요소는 메모리 관리 플래그의 어떤 상태에서도 제거되지 않습니다. 추출된 요소에 대한 포인터는 사용된 후 해제되어야 합니다.

예제:

```
//--- CList::DetachCurrent() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   CObject *object=list.DetachCurrent();
   if(object==NULL)
     {
      printf("분리 오류");
      delete list;
      return;
     }
   //--- 요소 사용
   //--- . . .
   //--- 요소 삭제
   delete object;
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteCurrent

리스트의 현재 위치에서 요소를 제거합니다.

```
bool  DeleteCurrent()
```

Return Value

성공시 true, 요소 제거 불가시 false.

Note

메모리 관리를 사용하도록 설정하면 제거된 요소의 메모리가 할당 취소됩니다.

예제:

```
//--- CList::DeleteCurrent() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   if(!list.DeleteCurrent())
     {
      printf("삭제 오류");
      delete list;
      return;
     }
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

리스트의 지정된 위치에서 요소를 제거합니다.

```
bool  Delete(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 목록에서 제거할 요소의 위치.

Return Value

성공시 true, 요소 제거 불가시 false.

Note

메모리 관리 플래그를 사용하도록 설정하면 삭제된 요소에 사용된 메모리가 해제됩니다.

예제:

```
//--- CList::Delete(int) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   if(!list.Delete(0))
     {
      printf("삭제 오류");
      delete list;
      return;
     }
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Clear

목록의 모든 요소를 제거합니다.

```
void  Clear()
```

Note

메모리 관리를 사용하도록 설정하면 삭제된 요소의 메모리가 할당 취소됩니다.

예제:

```
//--- CList::Clear() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   //--- 리스트 정리
   list.Clear();
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

IndexOf

지정된 리스트 요소의 색인을 가져옵니다.

```
int  IndexOf(
   CObject*  element      // 요소에 대한 포인터
   )
```

Parameters

element

[in] 목록 요소에 대한 포인터.

Return Value

목록 요소 색인, 또는 -1.

예제:

```
//--- CList::IndexOf(CObject*) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   CObject *object=new CObject;
   if(object==NULL)
     {
      printf("요소 생성 오류");
      delete list;
      return;
     }
   if(list.Add(object))
     {
      int pos=list.IndexOf(object);
     }
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetNodeAtIndex

리스트의 지정된 위치에서 요소를 가져옵니다.

```
CObject*  GetNodeAtIndex(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 목록에서의 요소 위치.

Return Value

요소에 대한 포인터 - 성공, Null - 포인터를 가져올 수 없음.

예제:

```
//--- CList::GetNodeAtIndex(int) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체 생성 오류");
      return;
     }
   //--- 목록 요소 추가
   //--- . . .
   CObject *object=list.GetNodeAtIndex(10);
   if(object==NULL)
     {
      printf("노드 오류 가져오기");
      delete list;
      return;
     }
   //--- 요소 사용
   //--- . . .
   //--- 요소를 삭제하지 마십시오
   //--- 목록 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetFirstNode

리스트의 첫 번째 요소를 가져옵니다.

```
CObject*  GetFirstNode()
```

Return Value

첫 번째 요소에 대한 포인터 - 성공, NULL - 포인터를 가져올 수 없음.

예제:

```
//--- CList::GetFirstNode() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   CObject *object=list.GetFirstNode();
   if(object==NULL)
     {
      printf("노드 오류 가져오기");
      delete list;
      return;
     }
   //--- 요소 사용
   //--- . . .
   //--- 요소를 삭제하지 마십시오
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetPrevNode

리스트의 이전 요소를 가져옵니다.

```
CObject*  GetPrevNode()
```

Return Value

이전 요소에 대한 포인터 - 성공, Null - 포인터를 가져올 수 없음.

예제:

```
//--- CList::GetPrevNode() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   CObject *object=list.GetPrevNode();
   if(object==NULL)
     {
      printf("노드 오류 가져오기");
      delete list;
      return;
     }
   //--- 요소 사용
   //--- . . .
   //--- 요소를 삭제하지 마십시오
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetCurrentNode

현재 리스트 요소를 가져옵니다.

```
CObject*  GetCurrentNode()
```

Return Value

현재 요소에 대한 포인터 - 성공, NULL - 포인터를 가져올 수 없음.

예제:

```
//--- CList::GetCurrentNode() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   CObject *object=list.GetCurrentNode();
   if(object==NULL)
     {
      printf("노드 오류 가져오기");
      delete list;
      return;
     }
   //--- 요소 사용
   //--- . . .
   //--- 요소를 삭제하지 마십시오
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetNextNode

리스트에서 다음 요소를 가져옵니다.

```
CObject*  GetNextNode()
```

Return Value

다음 요소에 대한 포인터 - 성공, Null - 포인터를 가져올 수 없음.

예제:

```
//--- CList::GetNextNode() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   CObject *object=list.GetNextNode();
   if(object==NULL)
     {
      printf("노드 오류 가져오기");
      delete list;
      return;
     }
   //--- 요소 사용
   //--- . . .
   //--- 요소를 삭제하지 마십시오
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

GetLastNode

리스트의 마지막 요소를 가져옵니다.

```
CObject*  GetLastNode()
```

Return Value

마지막 요소에 대한 포인터 - 성공, NULL - 포인터를 가져올 수 없음.

예제:

```
//--- CList::GetLastNode() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 리스트 요소를 추가합니다
   //--- . . .
   CObject *object=list.GetLastNode();
   if(object==NULL)
     {
      printf("노드 오류 가져오기");
      delete list;
      return;
     }
   //--- 요소 사용
   //--- . . .
   //--- 요소를 삭제하지 마십시오
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Sort

목록을 정렬합니다.

```
void  Sort(
   int  mode      // 저어렬 모드
   )
```

Parameters

mode

[in] 정렬 모드.

Return Value

No.

Note

목록은 항상 오름차순으로 정렬됩니다.

예제:

```
//--- CList::Sort(int) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 모드 0으로 정렬
   list.Sort(0);
   //--- 리스트 사용
   //--- ...
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

MoveToIndex

목록의 현재 요소를 지정된 위치로 이동합니다.

```
bool  MoveToIndex(
   int  pos      // 위치
   )
```

Parameters

pos

[in] 이동할 목록에 위치.

Return Value

성공적이면 true, 요소를 이동할 수 없으면 false.

예제:

```
//--- CList::MoveToIndex(int) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 시작할 현재 노드 이동
   list.MoveToIndex(0);
   //--- 리스트 사용
   //--- . . .
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Exchange

목록의 두 요소를 스왑합니다.

```
bool  Exchange(
   CObject*  node1,     // 목록 요소
   CObject*  node2      // 목록 요소
   )
```

Parameters

node1

[in] 목록 요소

node2

[in] 목록 요소

Return Value

성공시 true, 요소 스왑 불가시 false.

예제:

```
//--- CList::Exchange(CObject*, CObject*) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- exchange
   list.Exchange(list.GetFirstNode(),list.GetLastNode());
   //--- 리스트 사용
   //--- . . .
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

CompareList

목록을 다른 목록과 비교합니다.

```
bool  CompareList(
   CList*  list      // 소스에 대한 포인터
   )
```

Parameters

list

[in] 비교할 요소의 소스로 사용되는 CList 클래스의 인스턴스에 대한 포인터.

Return Value

true – 목록이 같음, false – 목록이 같지 않음.

예제:

```
//--- CList::CompareList(const CList*) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 소스 목록 생성
   CList *src=new CList;
   if(src==NULL)
     {
      printf("객체가 오류를 생성합니다");
      delete list;
      return;
     }
   //--- 소스 목록 채움
   //--- . . .
   //--- 다른 목록과 비교
   bool result=list.CompareList(src);
   //--- 목록 삭제
   delete src;
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Search

정렬된 리스트에서 샘플과 동일한 요소를 검색합니다.

```
CObject*  Search(
   CObject*  element      // 샘플
   )
```

Parameters

element

[in] 리스트에서 검색할 요소 샘플.

Return Value

찾은 요소에 대한 포인터 - 성공, NULL - 요소를 찾을 수 없음.

예제:

```
//--- CList::Search(CObject*) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 목록 요소 추가
   //--- . . .
   //--- 목록 정렬
   list.Sort(0);
   //--- 샘플 생성
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("샘플 생성 오류");
      delete list;
      return;
     }
   //--- 샘플 속성 설정
   //--- . . .
   //--- 요소 검색
   if(list.Search(sample)!=NULL) printf("요소가 검색됨");
   else                          printf("요소가 검색되지 않음");
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

목록 데이터를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen() 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료시 true, 그렇지 않으면 false.

예제:

```
//--- CList::Save(int) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 목록 요소 추가
   //--- . . .
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("File save: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 리스트 삭제
   delete list;
```

© 2000-2025, MetaQuotes Ltd.

```
   }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 목록 데이터를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen() 함수를 사용하여 연 이진 파일의 핸들.

Return Value

성공적으로 완료시 true, 그렇지 않으면 false.

Note

파일에서 목록 요소를 읽을 때 CList::CreateElement(int) 메서드를 호출하여 각 요소를 만듭니다.

예제:

```
//--- CLoad::Load(int) 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 파일 오픈
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("File load: Error %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 목록 요소 사용
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- . . .
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

리스트 유형 식별자를 가져옵니다.

```
virtual int  Type()
```

Return Value

목록 유형 식별자(CList -7779용).

예제:

```
//--- CList::Type() 예제
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("객체가 오류를 생성합니다");
      return;
     }
   //--- 목록 유형 가져오기
   int type=list.Type();
   //--- 리스트 삭제
   delete list;
  }
```

© 2000-2025, MetaQuotes Ltd.

CTreeNode

CTreeNode는 CTree 이진 트리 노드의 클래스입니다.

Description

CTreeNode는 CTree 이진 트리의 노드로 작업할 수 있는 기능을 제공합니다. 트리를 통한 탐색 옵션은 클래스에서 구현됩니다. 또한 파일로 작업하는 방법도 구현됩니다.

Declaration

```
class CTreeNode : public CObject
```

Title

```
#include <Arrays\TreeNode.mqh>
```

상속 계층

CObject

CTreeNode

직계 종속

CTree

그룹별 클래스 메서드

Attributes	
Owner	소유자 노드의 포인터를 가져오거나 설정합니다
Left	왼쪽 노드의 포인터를 가져오거나 설정합니다
Right	오른쪽 노드의 포인터를 가져오거나 설정합니다
Balance	노드 밸런스를 가져옵니다
BalanceL	노드의 왼쪽 하위 분기의 밸런스를 가져옵니다
BalanceR	노드의 오른쪽 하위 분기의 균형을 가져옵니다
새 요소 작성	
CreateSample	새 노드 인스턴스를 작성합니다
Comparison	
RefreshBalance	노드 밸런스를 다시 계산합니다
Search	
GetNext	다음 노드의 포인터를 가져옵니다
Input/Output	
SaveNode	노드 데이터를 파일에 저장합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
LoadNode	파일에서 노드 데이터를 다운로드합니다
virtual Type	노드 유형의 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Compare

CTreeNode 클래스 하위 트리는 실제 응용이 가능하다.

CTreeNode 클래스의 하위 항목에는 미리 정의된 메서드가 있어야 합니다: CreateSample은 CTreeNode 하위 클래스의 새 인스턴스를 생성하고, Compare는 CTreeNode의 하위 클래스, Type(노드 식별이 필요한 경우), SaveNode 그리고 LoadNode(파일로 작업해야 하는 경우)의 키 필드 값을 비교합니다.

CTree 하위 클래스의 예를 들어 보겠습니다.

```
//+------------------------------------------------------------------+
//|                                                   MyTreeNode.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             www.metaquotes.net   |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+------------------------------------------------------------------+
//| CTreeNode에서 파생된 CMyTreeNode 클래스를 설명합니다.              |
//+------------------------------------------------------------------+
//| Class CMyTreeNode.                                               |
//| 목적: 이진 트리의 요소 클래스.                                     |
//|              클래스 CTreeNode의 하위 항목.                         |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- 이용자 데이터
   long              m_long;           // long 유형의 필드
   double            m_double;         // double 유형의 커스텀 변수
   string            m_string;         // string 유형의 커스텀 변수
   datetime          m_datetime;       // datetime 유형의 커스텀 변수

public:
                     CMyTreeNode();
   //--- 사용자 데이터에 액세스하는 방법
   long              GetLong(void)               { return(m_long); }
   void              SetLong(long value)         { m_long=value;   }
   double            GetDouble(void)             { return(m_double); }
   void              SetDouble(double value)     { m_double=value;   }
   string            GetString(void)             { return(m_string); }
```

© 2000-2025, MetaQuotes Ltd.

```
   void              SetString(string value)       { m_string=value;   }
   datetime          GetDateTime(void)             { return(m_datetime); }
   void              SetDateTime(datetime value)   { m_datetime=value;   }
   //--- 파일 작업 방법
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- 클래스 인스턴스 생성 방법
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| CMyTreeNode 클래스 구성자.                                       |
//| INPUT:  없음.                                                    |
//| OUTPUT: none.                                                    |
//| 비고: 없음.                                                      |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- 사용자 데이터 초기화
   m_long        =0;
   m_double      =0.0;
   m_string      ="";
   m_datetime    =0;
  }
//+------------------------------------------------------------------+
//| 지정된 알고리즘으로 다른 트리 노드와 비교.                       |
//| INPUT:  노드 - 비교할 트리 요소,                                 |
//|         모드 - 비교 알고리즘의 식별자.                          |
//| OUTPUT: 비교 결과 (>0, 0, <0).                                   |
//| 비고: 없음.                                                      |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- 트리 생성 알고리즘이 유일하기 때문에 모드 매개 변수가 무시됩니다
   int res=0;
//--- 명시적 유형 캐스팅
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| 새 클래스 인스턴스 만들기.                                       |
//| INPUT:  없음.                                                    |
//| OUTPUT: CMyTreeNode 클래스의 새 인스턴스에 대한 포인터.          |
//| 비고: 없음.                                                      |
//+------------------------------------------------------------------+
CTreeNode* CMyTreeNode::CreateSample()
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
  }
//+------------------------------------------------------------------+
//| 파일에 트리 노드 데이터를 씁니다.                                |
//| INPUT:  file_handle -쓰기 위해 미리 열린 파일의 핸들.    |
//| OUTPUT: 정상이면 true, 그렇지 않으면 false입니다.                |
//| 비고: 없음.                                                      |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- 점검
   if(file_handle<0) return(false);
//--- 사용자 데이터 쓰기
//--- long 유형의 커스텀 변수 쓰기
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))          return(false);
//--- double 유형의 커스텀 변수 쓰기
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))    return(false);
//--- 문자열 유형의 커스텀 변수 쓰기
   len=StringLen(m_string);
//--- 문자열 길이 작성
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)   return(false);
//--- 문자열 작성
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- datetime 유형의 커스텀 변수 작성
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))      return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| 파일에서 트리 노드 데이터를 읽습니다.                            |
//| INPUT:  file_handle -읽기 위해 미리 열어 놓은 파일의 핸들.    |
//| OUTPUT: 정상이면 true, 그렇지 않으면 false입니다.                |
//| 비고: 없음.                                                      |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- 점검
   if(file_handle<0) return(false);
//--- 읽기
   if(FileIsEnding(file_handle)) return(false);
//--- char 타입의 커스텀 변수 읽기
//--- long 타입의 커스텀 변수 읽기
   m_long=FileReadLong(file_handle);
//--- double 타입의 커스텀 변수 읽기
```

© 2000-2025, MetaQuotes Ltd.

```
   m_double=FileReadDouble(file_handle);
//--- string 타입의 커스텀 변수 읽기
//--- string 길이 읽기
   len=FileReadInteger(file_handle,INT_VALUE);
//--- string 읽기
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- datetime 유형의 커스텀 변수 읽기
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

Owner

소유자 노드의 포인터를 가져옵니다.

```
CTreeNode*  Owner()
```

Return Value

소유자 노드의 포인터.

Owner

소유자 노드의 포인터를 설정합니다.

```
void  Owner(
   CTreeNode*  node      // 노드
   )
```

Parameters

node

[in] 소유자 노드의 포인터 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Left

왼쪽 노드의 포인터를 가져옵니다.

```
CTreeNode*  Left()
```

Return Value

왼쪽 노드의 포인터.

Left

왼쪽 노드의 포인터를 설정합니다.

```
void  Left(
   CTreeNode*  node      // 노드
   )
```

Parameters

node

[in] 왼쪽 노드의 포인터의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Right

오른쪽 노드의 포인터를 가져옵니다.

```
CTreeNode*  Right()
```

Return Value

오른쪽 노드의 포인터.

Right

오른쪽 노드의 포인터를 설정합니다.

```
void  Right(
   CTreeNode*  node      // 노드
   )
```

Parameters

node

[in] 오른쪽 노드의 포인터의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Balance

노드 밸런스를 가져옵니다.

```
int  Balance() const
```

Return Value

노드 밸런스.

© 2000-2025, MetaQuotes Ltd.

BalanceL

노드의 왼쪽 하위 분지의 밸런스를 가져옵니다.

```
int  BalanceL() const
```

Return Value

노드의 왼쪽 하위 분기 균형.

© 2000-2025, MetaQuotes Ltd.

BalanceR

노드의 오른쪽 하위 분기의 균형을 가져옵니다.

```
int  BalanceR() const
```

Return Value

노드의 오른쪽 하위 분기 균형.

© 2000-2025, MetaQuotes Ltd.

CreateSample

새 노드 샘플을 만듭니다.

```
virtual CTreeNode*  CreateSample()
```

Return Value

새 노드 샘플 또는 NULL에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

RefreshBalance

노드 밸런스를 다시 계산합니다.

```
int  RefreshBalance()
```

Return Value

노드 밸런스.

© 2000-2025, MetaQuotes Ltd.

GetNext

다음 노드의 포인터를 가져옵니다.

```
CTreeNode*  GetNext(
   CTreeNode*  node      // 노드
   )
```

Parameters

node

[in] 검색 시작 노드.

Return Value

다음 노드의 포인터.

© 2000-2025, MetaQuotes Ltd.

SaveNode

파일에 노드 데이터를 씁니다.

```
bool  SaveNode(
   int  file_handle      // 핸들
   )
```

Parameters

file_handle

[in] 쓰기 위해 이전에 열었던 이진 파일의 핸들.

Return Value

true - 성공, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

LoadNode

파일에서 노드 데이터를 읽습니다.

```
bool  LoadNode(
   int          file_handle,     // 핸들
   CTreeNode*   main             // 노드
   )
```

Parameters

file_handle

[in] 읽기 위해 이전에 열었던 이진 파일의 핸들.

main

[in] 데이터용 노드.

Return Value

true – 성공, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Type

노드 유형의 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

노드 유형의 식별자.

© 2000-2025, MetaQuotes Ltd.

CTree

CTree는 CTreeNode 클래스 및 그 하위 항목 인스턴스의 이진 트리 클래스입니다.

Description

CTree 클래스는 CTreeNode 클래스 인스턴스와 하위 인스턴스의 이진 트리로 작업할 수 있는 가능성을 제공합니다. 트리 요소를 추가/삽입/삭제하고 트리에서 검색하는 옵션이 클래스에 구현됩니다. 그 외에도 파일로 작업하는 방식이 구현됩니다.

동적 메모리 관리 메커니즘은 CTree 클래스에 구현되지 않습니다(CList 및 CArrayObj클래스와는 달리). 메모리 할당 해제를 통해 모든 트리 노드가 삭제됩니다.

Declaration

```
class CTree : public CTreeNode
```

Title

```
#include <Arrays\Tree.mqh>
```

상속 계층

CObject
  CTreeNode
    CTree

그룹별 클래스 메서드

Attributes	
Root	트리의 루트 노드를 가져옵니다.
새 요소 작성	
CreateElement	노드의 새 인스턴스를 작성합니다
Filling	
Insert	트리에 노드를 추가합니다
Deletion	
Detach	트리에서 지정된 노드를 분리합니다
Delete	트리에서 지정된 노드를 삭제합니다
Clear	트리의 모든 노드를 삭제합니다
Search	
Find	샘플별로 트리에서 노드를 검색합니다
Input/output	
virtual Save	모든 트리 데이터를 파일에 저장합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
virtual Load	파일에서 트리 데이터를 다운로드합니다
virtual Type	트리 유형의 식별자를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CTreeNode 클래스에서 상속된 메서드

Parent, Parent, Left, Left, Right, Right, Balance, BalanceL, BalanceR, RefreshBalance, GetNext, SaveNode, LoadNode

CTreeNode 클래스 하위 항목(클래스 CTree의 하위 항목)의 트리는 실제 적용 대상이 됩니다.

CTree 클래스의 하위 항목에는 미리 정의된 메서드 CreateElement - CTreeNode 하위 클래스의 새 인스턴스를 생성하는-가 있어야합니다.

CTree 하위 클래스의 예를 들어 보겠습니다.

```
//+------------------------------------------------------------------+
//|                                                      MyTree.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//| www.metaquotes.net |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+------------------------------------------------------------------+
//| CTree에서 파생된 CMyTree 클래스를 설명합니다.                          |
//+------------------------------------------------------------------+
//| Class CMyTree.                                                   |
//| 목적: 이진 검색 트리의 구성 및 탐색.     |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- 사용자 정의 데이터로 트리에서 검색하는 방법
   CMyTreeNode*         FindByLong(long find_long);
   //--- 트리 요소 생성 방법
   virtual CTreeNode *CreateElement();
  };
//---
CMyTree MyTree;
//+------------------------------------------------------------------+
//| 새 트리 노드의 생성.                                              |
```

© 2000-2025, MetaQuotes Ltd.

```
//| INPUT:  없음.                                                    |
//| OUTPUT: 새 트리 노드에 대한 포인터 또는 NULL.                    |
//| 비고: 없음.                                                      |
//+------------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+------------------------------------------------------------------+
//| m_long 값을 사용하여 목록에서 요소 검색.                         |
//| INPUT:  find_long - 검색된 값.                                   |
//| OUTPUT: 찾은 목록 요소의 포인터, 또는 NULL.                      |
//| 비고: 없음.                                                      |
//+------------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- 검색 매개 변수를 전달하기 위한 트리 노드 생성
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| script "testing of class CMyTree"                                |
//+------------------------------------------------------------------+
//---  문자열 초기화를 위한 배열
string str_array[11]={"p","oo","iii","uuuu","yyyyy","ttttt","rrrr","eee","ww","q","999
//---
int OnStart() export
  {
   int          i;
   uint         pos;
   int          beg_time,end_time;
   CMyTreeNode *node; //--- 클래스 CMyTreeNode 샘플에 대한 임시 포인터
//---
   printf("Start test %s.",__FILE__);
//--- extCountedNodes의 양으로 MyTreeNode 클래스의 인스턴스를 채웁니다.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i++)
     {
      node=MyTree.CreateElement();
```

© 2000-2025, MetaQuotes Ltd.

```
      if(node==NULL)
        {
         //--- 비상구
         printf("%s (%4d): create error",__FILE__,__LINE__);
         return(__LINE__);
        }
      NodeSetData(node,i);
      node.SetLong(i);
      MyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("MyTree 채우기 시간은 %d ms.",end_time-beg_time);
//--- 임시 트리 TmpMyTree 생성.
   CMyTree TmpMyTree;
//--- 트리 요소의 50% 분리(모두 짝수)
//--- 임시 트리 TmpMyTree에 추가.
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i+=2)
     {
      node=MyTree.FindByLong(i);
      if(node!=NULL)
         if(MyTree.Detach(node)) TmpMyTree.Insert(node);
     }
   end_time=GetTickCount();
   printf("MyTree %d 요소 삭제시간은 %d ms.",extCountedNodes/2,end_time-beg_time);
//--- 분리 반환
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Save(int file_handle) 메서드의 작업 확인;
   int file_handle;
   file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!MyTree.Save(file_handle))
        {
         //--- 파일에 쓰는 중 오류 발생
         //--- 비상구
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- 떠나기 전에 파일을 닫으세요!!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//--- Load(int file_handle) 메서드의 작업 확인;
```

© 2000-2025, MetaQuotes Ltd.

```
   file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- 파일에서 읽는 중 오류 발생
         //--- 비상구
         printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
         //--- 떠나기 전에 파일을 닫으세요!!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("테스트 종료 %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| 노드 내용을 저널로 출력하는 함수                                |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
              node.GetLong(),node.GetDouble(),
              node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| 노드를 임의 값으로 채우는 함수                                   |
//+------------------------------------------------------------------+
void NodeSetData(CMyTreeNode *node,int mode)
  {
   if(mode%2==0)
     {
      node.SetLong(mode*MathRand());
      node.SetDouble(MathPow(2.02,mode)*MathRand());
     }
   else
     {
      node.SetLong(mode*(long)(-1)*MathRand());
      node.SetDouble(-MathPow(2.02,mode)*MathRand());
     }
   node.SetString(str_array[mode%10]);
   node.SetDateTime(10000*mode);
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Root

트리의 루트 노드를 가져옵니다.

```
CTreeNode*  Root() const
```

Return Value

트리의 루트 노드 포인터.

© 2000-2025, MetaQuotes Ltd.

CreateElement

노드의 새 인스턴스를 만듭니다.

```
virtual CTreeNode*  CreateElement()
```

Return Value

노드 또는 NULL의 새 인스턴스 포인터.

© 2000-2025, MetaQuotes Ltd.

Insert

트리에 노드를 추가합니다.

```
CTreeNode*  Insert(
   CTreeNode*  new_node      // 노드
   )
```

Parameters

new_node

[in] 트리에 삽입할 노드의 포인터.

Return Value

소유자 노드 또는 NULL의 포인터.

© 2000-2025, MetaQuotes Ltd.

Detach

트리에서 지정한 노드를 분리합니다.

```
bool  Detach(
   CTreeNode*  node      // 노드
   )
```

Parameters

node

[in] 분리할 노드 포인터.

Return Value

true - 성공, 그렇지 않으면 false.

Note

분리 후 노드 포인터가 해제되지 않습니다. 그 나무는 균형이 잡혀 있다.

© 2000-2025, MetaQuotes Ltd.

Delete

트리에서 지정한 노드를 삭제합니다.

```
bool  Delete(
   CTreeNode*  node      // 노드
   )
```

Parameters

node

[in] 삭제할 노드 포인터.

Return Value

true – 성공, 그렇지 않으면 false.

Note

삭제 후 노드 포인터가 해제됩니다. 그 나무는 균형이 잡혀 있다.

© 2000-2025, MetaQuotes Ltd.

Clear

트리의 모든 노드를 삭제합니다.

```
void  Clear()
```

Return Value

None.

Note

삭제 후 노드 포인터가 릴리스됩니다.

© 2000-2025, MetaQuotes Ltd.

Find

트리에서 노드를 샘플별로 검색합니다.

```
CTreeNode*  Find(
   CTreeNode*  node      // 노드
   )
```

Parameters

node

[in] 검색 샘플로 사용되는 데이터가 포함된 노드.

Return Value

발견된 노드 또는 NULL의 포인터.

© 2000-2025, MetaQuotes Ltd.

Save

파일에 트리 데이터를 씁니다.

```
virtual bool  Save(
   int  file_handle      // 핸들
   )
```

Parameters

file_handle

[in] 쓰기 위해 이전에 열었던 이진 파일의 핸들.

Return Value

true - 성공, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 트리 데이터를 읽습니다.

```
virtual bool  Load(
   int  file_handle      // 핸들
   )
```

Parameters

file_handle

[in] 읽기 위해 이전에 열었던 이진 파일의 핸들.

Return Value

true - 성공, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Type

트리 유형의 식별자를 가져옵니다.

```
virtual int  Type() const
```

Return Value

트리 유형의 식별자.

© 2000-2025, MetaQuotes Ltd.

일반 데이터 모음

라이브러리는 일반 컬렉션을 정의하는 클래스 및 인터페이스를 제공하여 사용자가 강력한 유형의 컬렉션을 생성할 수 있습니다. 이러한 컬렉션은 일반적이지 않은 유형 컬렉션보다 더 편리하고 데이터 처리 성능을 제공합니다.

라이브러리는 터미널 작업 디렉토리의 Include\Generic 폴더에서 사용할 수 있습니다.

Objects:

개체	설명	Type
ICollection	일반 데이터 수집을 구현하기 위한 인터페이스	INTERFACE
IEqualityComparable	비교할 수 있는 개체를 구현하기 위한 인터페이스	INTERFACE
IComparable	"크거나 작거나 같음" 측면에서 비교할 수 있는 개체 구현을 위한 인터페이스	INTERFACE
IComparer	한 개체가 다른 개체보다 크거나 작거나 같은지 여부에 관계없이 T 유형의 두 개체를 비교하는 일반 클래스를 구현하기 위한 인터페이스	INTERFACE
IEqualityComparer	동일성을 위해 T 유형의 두 개체를 비교하는 일반 클래스를 구현하기 위한 인터페이스	INTERFACE
IList	일반 데이터 목록 구현을 위한 인터페이스	INTERFACE
IMap	키/값 페어의 일반 컬렉션을 구현하기 위한 인터페이스	INTERFACE
ISet	일반 데이터 집합 구현을 위한 인터페이스	INTERFACE
CDefaultComparer	Compare 글로벌 메서드를 기반으로 IComparer<T> 일반 인터페이스를 구현하는 도우미 클래스입니다	CLASS
CDefaultEqualityComparer	Equals<T> 및 GetHashCode 글로벌 메서드를 사용하여 IEqualityComparer<T> 일반 인터페이스를 구현하는 도우미 클래스	CLASS
CArrayList	IList<T> 인터페이스를 구현하는 일반 클래스	CLASS
CKeyValuePair	클래스는 키/값 페어를 구현합니다	CLASS
CHashMap	IMap<TKey, TValue> 인터페이스를 구현하는 일반 클래스	CLASS

© 2000-2025, MetaQuotes Ltd.

개체	설명	Type
CHashSet	ISet<T> 인터페이스를 구현하는 일반 클래스	CLASS
CLinkedListNode	CLinkedListNode<T> 클래스를 구현하기 위한 도우미 클래스	CLASS
CLinkedList	ICollection<T> 인터페이스를 구현하는 일반 클래스	CLASS
CQueue	ICollection<T> 인터페이스를 구현하는 일반 클래스	CLASS
CRedBlackTreeNode	CRedBlackTree<T> 클래스를 구현하는 데 사용되는 도우미 클래스	CLASS
CRedBlackTree	ICollection<T> 인터페이스를 구현하는 일반 클래스	CLASS
CSortedMap	IMap<TKey, TValue> 인터페이스를 구현하는 일반 클래스	CLASS
CSortedSet	ISet<T> 인터페이스를 구현하는 일반 클래스	CLASS
CStack	ICollection<T> 인터페이스를 구현하는 일반 클래스	CLASS

Global methods:

메서드	설명
ArrayBinarySearch	요소를 비교하기 위해 IComparable<T> 인터페이스를 사용하여 오름차순 1차원 배열에서 지정된 값을 검색
ArrayIndexOf	1차원 배열에서 값이 처음 나타나는 경우 검색
ArrayLastIndexOf	1차원 배열에서 값이 마지막으로 발생했는지 검색
ArrayReverse	1차원 배열의 요소 시퀀스를 변경
Compare	두 값 중 하나가 다른 값보다 크거가 작거나 같은지 여부 비교
Equals	동일성에 대한 두 값 비교
GetHashCode	해시 코드 값을 계산

© 2000-2025, MetaQuotes Ltd.

ICollection<T>

ICollection<T>는 일반 데이터 수집을 구현하기 위한 인터페이스.

설명

ICollection<T> 인터페이스는 요소를 세는 방법, 컬렉션을 지우는 방법, 요소를 추가하거나 삭제하는 방법을 등을 포함하여 컬렉션과 함께 작업할 기본 방법을 결정합니다.

선언

```
template<typename T>
interface ICollection
```

헤더

```
#include <Generic\Interfaces\ICollection.mqh>
```

상속 계층

ICollection

직계 후손

CLinkedList, CQueue, CRedBlackTree, CStack, IList, IMap, ISet

클래스 메서드

메서드	설명
Add	컬렉션에 추가할 요소
Count	컬렉션의 요소 수를 반환
Contains	컬렉션에 지정된 값의 요소가 포함되어 있는지 여부를 결정
CopyTo	지정한 인덱스에서 시작하여 지정된 배열에 컬렉션의 모든 요소를 복사
Clear	컬렉션에서 모든 요소를 제거
Remove	컬렉션에서 지정된 요소의 첫 번째 항목을 제거

© 2000-2025, MetaQuotes Ltd.

Add

컬렉션에 추가할 요소.

```
bool Add(
   T  value     // 요소의 값
   );
```

매개변수

값

[in] 추가할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

컬렉션의 요소 수를 반환.

```
int Count();
```

값 반환

요소 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

컬렉션에 지정된 값의 요소가 포함되어 있는지 여부를 결정.

```
bool Contains(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

지정한 값을 가진 요소가 컬렉션에 있으면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CopyTo

컬렉션의 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사.

```
int CopyTo(
   T&          dst_array[],     // 표기 배열
   const int   dst_start=0      // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 컬렉션 요소를 표기할 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 요소 수를 반환

© 2000-2025, MetaQuotes Ltd.

Clear

모음의 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

컬렉션에서 지정한 요소의 첫 번째 항목을 제거.

```
bool Remove(
   T  item     // 요소 값
   );
```

매개변수

항목

[in] 삭제할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IEqualityComparable<T>

IEqualityComparable<T>는 비교할 수 있는 개체를 구현하기 위한 인터페이스입니다.

설명

IEqualityComparable<T> 인터페이스는 현재 개체의 해시 코드를 검색하고 동일한 유형의 다른 개체와 동일한지 여부를 확인하는 메서드를 정의합니다.

선언

```
template<typename T>
interface IEqualityComparable
```

헤더

```
#include <Generic\Interfaces\IEqualityComparable.mqh>
```

상속 계층

IEqualityComparable

직계 후손

IComparable

클래스 메서드

메서드	설명
Equals	현재 개체를 지정된 값과 비교
HashCode	현재 개체의 해시 코드 값을 계산

© 2000-2025, MetaQuotes Ltd.

Equals

현재 개체를 지정된 값과 비교.

```
bool Equals(
   T  value     // 비교할 값
   );
```

매개변수

값

[in] 현재 개체를 비교할 값.

값 반환

개체가 같으면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

HashCode

현재 개체의 해시 코드 값을 계산.

```
int HashCode();
```

값 반환

해시 코드 반환.

© 2000-2025, MetaQuotes Ltd.

IComparable<T>

IComparable<T>는 한 개체가 다른 개체보다 크거나 작거나 같은지 알아보기 위해 비교할 수 있는 개체를 구현하기 위한 인터페이스입니다

설명

IComparable<T> 인터페이스는 이러한 개체의 컬렉션을 정렬할 수 있는 기준으로 현재 개체를 동일한 유형의 다른 개체와 비교하는 방법을 정의합니다.

선언

```
template<typename T>
interface IComparable : public IEqualityComparable<T>
```

헤더

```
#include <Generic\Interfaces\IComparable.mqh>
```

상속 계층

IEqualityComparable

IComparable

직계 후손

CKeyValuePair

클래스 메서드

메서드	설명
Compare	현재 개체를 지정된 값과 비교

© 2000-2025, MetaQuotes Ltd.

Compare

현재 개체를 지정된 값과 비교.

```
int Compare(
   T  value     // 비교할 값
   );
```

매개변수

값

[in] 현재 개체를 비교할 값.

값 반환

현재 개체와 전달된 개체의 비율을 나타내는 숫자를 반환:

- 결과가 0보다 작으면 현재 개체는 전달된 개체보다 작습니다
- 결과가 0이면 현재 개체는 전달된 개체와 같습니다
- 결과가 0보다 크면 현재 개체는 전달된 개체보다 큽니다

© 2000-2025, MetaQuotes Ltd.

IComparer<T>

IComparer<T>는 T 유형의 두 개체를 비교하는 일반적인 클래스를 구현하기 위한 인터페이스입니다

설명

IComparer<T> 인터페이스는 이러한 개체들이 집합이 정렬될 수 있는 기초에 기초하여 T 유형의 두 개체를 비교하는 방법을 결정합니다.

선언

```
template<typename T>
interface IComparer
```

헤더

```
#include <Generic\Interfaces\IComparer.mqh>
```

상속 계층

IComparer

직계 후손

CDefaultComparer

클래스 메서드

메서드	설명
Compare	T 유형의 두 값 비교하기

© 2000-2025, MetaQuotes Ltd.

Compare

T 유형의 두 값 비교하기.

```
int Compare(
   T  x,      // 첫 번째 값
   T  y       // 두 번째 값
   );
```

매개변수

x

[in] 비교할 첫 번째 값.

y

[in] 비교할 첫 번째 값.

값 반환

비교된 두 값의 비율을 나타내는 숫자를 반환:

- 결과가 0보다 작으면, x가 y보다 작음 (x<y)
- 결과가 0이면, x가 y와 같음 (x=y)
- 결과가 0보다 크면, x가 y보다 큼 (x>y)

© 2000-2025, MetaQuotes Ltd.

IEqualityComparer<T>

IEqualityComparer<T>는 T 유형의 두 개체를 비교하는 일반 클래스를 구현하기 위한 인터페이스입니다.

설명

IEqualityComparer<T> 인터페이스는 T 유형 개체의 해시 코드를 검색하고 T 유형 개체의 두 개체가 동일한지 확인하는 방법을 정의합니다.

선언

```
template<typename T>
interface IEqualityComparer
```

헤더

```
#include <Generic\Interfaces\IEqualityComparer.mqh>
```

상속 계층

IEqualityComparer

직계 후손

CDefaultEqualityComparer

클래스 메서드

메서드	설명
Equals	T 유형의 두 값 비교하기
HashCode	T 유형 개체를 기준으로 해시 코드 값을 계산

© 2000-2025, MetaQuotes Ltd.

Equals

T 유형의 두 값 비교하기.

```
bool Equals(
   T  x,      // 첫 번째 값
   T  y       // 두 번째 값
   );
```

매개변수

x

[in] 비교할 첫 번째 값.

y

[in] 비교할 두 번째 값.

값 반환

값이 같으면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

HashCode

T 유형 개체를 기준으로 해시 코드 값을 값을 계산.

```
int HashCode(
   T  value     // 계산할 개체
   );
```

매개변수

값

[in] 해시 코드를 가져올 개체.

값 반환

해시 코드 반환.

© 2000-2025, MetaQuotes Ltd.

IList<T>

IList<T>는 일반 데이터 목록을 구현하기 위한 인터페이스입니다.

설명

IList<T> 인터페이스는 인덱스별로 요소에 액세스하고 요소를 검색 및 삭제하며 정렬하는 등의 목록 작업을 위한 기본 방법을 정의합니다.

선언

```
template<typename T>
interface IList : public ICollection<T>
```

헤더

```
#include <Generic\Interfaces\IList.mqh>
```

상속 계층

ICollection

IList

직계 후손

CArrayList

클래스 메서드

메서드	설명
TryGetValue	지정한 인덱스에서 목록 요소를 가져오기
TrySetValue	지정한 인덱스의 목록에서 값을 변경하기
Insert	지정된 인덱스의 목록에 요소를 삽입하기
IndexOf	목록에서 값이 처음 나타나는 경우 검색하기
LastIndexOf	목록에서 값이 마지막으로 발생했는지 검색하기
RemoveAt	지정한 인덱스에서 목록 요소를 제거하기

© 2000-2025, MetaQuotes Ltd.

TryGetValue

지정한 인덱스에서 목록 요소를 가져오기.

```
bool TryGetValue(
   const int  index,      // 요소 인덱스
   T&         value       // 표기 변수
   );
```

매개변수

인덱스

[in] 목록의 요소 인덱스.

&value

[out] 목록에서 요소의 지정된 값을 표기할 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

지정한 인덱스의 목록에서 값을 변경하기.

```
bool TrySetValue(
   const int  index,      // 요소 인덱스
   T          value       // 새 값
   );
```

매개변수

인덱스

[in] 목록의 요소 인덱스.

값

[in] 지정된 목록 요소에 할당할 새 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Insert

지정된 인덱스의 목록에 요소를 삽입하기.

```
bool Insert(
   const int  index,      // 삽입할 인덱스
   T          item        // 삽입할 값
   );
```

매개변수

인덱스

[in] 삽입할 인덱스.

항목

[in] 지정된 인덱스에 삽입될 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IndexOf

목록에서 값이 처음 나타나는 경우를 검색.

```
int IndexOf(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

처음 찾은 요소의 인덱스를 반환. 값을 찾을 수 없으면 -1을 반환.

© 2000-2025, MetaQuotes Ltd.

LastIndexOf

목록에서 값이 마지막으로 표시되는 항목을 검색.

```
int LastIndexOf(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

마지막으로 찾은 요소의 인덱스를 반환. 값을 찾을 수 없으면 -1을 반환.

© 2000-2025, MetaQuotes Ltd.

RemoveAt

지정한 인덱스에서 목록 요소를 제거하기.

```
bool RemoveAt(
   const int  index     // 요소 인덱스
   );
```

매개변수

인덱스

[in] 삭제할 요소의 인덱스.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IMap<TKey, TValue>

IMap<TKey, TValue>는 키/값 페어의 일반 컬렉션을 구현하기 위한 인터페이스입니다.

설명

IMap<TKey, TValue> 인터페이스는 데이터가 키/값 페어로 저장된 컬렉션에 대해 작업하는 기본 방법을 정의합니다.

선언

```
template<typename TKey, typename TValue>
interface IMap : public ICollection<TKey>
```

헤더

```
#include <Generic\Interfaces\IMap.mqh>
```

상속 계층

ICollection

IMap

직계 후손

CHashMap, CSortedMap

클래스 메서드

메서드	설명
Add	컬렉션에 키/값 페어를 추가하기
Contains	컬렉션에 지정된 키가 있는 키/값 테이블이 보함되어 있는지 여부를 결정
Remove	컬렉션에서 키/값 페어의 첫 번째 항목을 제거하기
TryGetValue	집합에서 지정된 키를 가진 요소를 가져오기
TrySetValue	지정된 키의 컬렉션에서 키/값 페어 값을 변경하기
CopyTo	지정된 인덱스에서 시작하여 컬렉션의 모든 키/값 페어를 지정된 배열로 복사하기

© 2000-2025, MetaQuotes Ltd.

Add

컬렉션에 키/값 페어를 추가하기.

```
bool Add(
   TKey    key,       // 키
   TValue  value      // 값
   );
```

매개변수

키

[in] 키.

값

[in] 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

컬렉션에 지정된 키/값 테이블이 포함되어 있는지 여부를 결정.

```
bool Contains(
   TKey    key,       // 키
   TValue  value      // 값
   );
```

매개변수

키

[in] 키.

값

[in] 값.

값 반환

컬렉션에 지정된 키 및 키/값 페어가 포함되어 있으면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Remove

컬렉션에서 키/값 페어의 첫 번째 항목을 제거하기.

```
bool Remove(
   TKey  key      // 키
   );
```

매개변수

키

[in] 키.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

컬렉션에서 지정한 키를 가진 요소를 가져오기.

```
bool TryGetValue(
   TKey      key,       // 키
   TValue&   value      // 값 표기 변수
   );
```

매개변수

키

[in] 키.

&value

[out] 키/값 페어의 지정된 값을 쓸 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

지정된 키의 컬렉션에서 키/값 페어의 값을 변경하기.

```
bool TrySetValue(
   TKey    key,       // 키
   TValue  value      // 새 값
   );
```

매개변수

키

[in] 키.

값

[in] 지정된 키-값 페어에 할당할 새 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CopyTo

지정된 인덱스에서 시작하여 컬렉션의 모든 키/값 페어를 지정된 배열로 복사하기.

```
int CopyTo(
   TKey&       dst_keys[],       // 키 표기 배열
   TValue&     dst_values[],     // 값 표기 베열
   const int   dst_start=0       // 표기 시작 인덱스
   );
```

매개변수

&dst_keys[]

[out] 컬렉션의 모든 키를 기록할 배열.

&dst_values[]

[out] 컬렉션의 해당 키 값을 표기할 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 키/값 페어의 수를 반환.

© 2000-2025, MetaQuotes Ltd.

ISet<T>

ISet<T>는 일반 데이터 집합을 구현하기 위한 인터페이스.

설명

ISet 인터페이스는 집합의 합집합과 교집합, 완전 및 비완전 부분집합 정의 등과 같은 집합과 관련된 기본 방법을 정의합니다.

선언

```
template<typename T>
interface ISet : public ICollection<T>
```

헤더

```
#include <Generic\Interfaces\ISet.mqh>
```

상속 계층

ICollection

ISet

직계 후손

CHashSet, CSortedSet

클래스 메서드

메서드	설명
ExceptWith	현재 컬렉션과 통과된 컬렉션(배열) 간의 차이 연산을 생성
IntersectWith	현재 컬렉션과 통과된 컬렉션(배열)의 교차점 연산을 생성
SymmetricExceptWith	현재 컬렉션과 통과된 컬렉션(배열) 간의 대칭 차이 연산을 생성
UnionWith	현재 컬렉션과 통과된 컬렉션(배열)의 결합을 생성
IsProperSubsetOf	현재 집합이 지정된 컬렉션 또는 배열의 적절한 부분 집합인지 여부를 결정
IsProperSupersetOf	현재 집합이 지정된 컬렉션 또는 배열의 적절한 상위 집합인지 여부를 결정
IsSubsetOf	현재 집합이 지정된 컬렉션 또는 배열의 하위 집합인지 여부를 결정
IsSupersetOf	현재 집합이 지정된 컬렉션 또는 배열의 상위 집합인지 여부를 결정
Overlaps	현재 집합이 지정된 컬렉션 또는 배열과 겹치는지 여부를 결정
SetEquals	현재 집합에 지정된 컬렉션 또는 배열의 모든 요소가 포함되어 있는지 여부를 결정

© 2000-2025, MetaQuotes Ltd.

ExceptWith

현재 컬렉션과 통과된 컬렉션(배열) 간의 차이 연산을 생성. 지정된 컬렉션(배열)에 있는 모든 요소를 현재 컬렉션(배열)에서 제거.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void ExceptWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void ExceptWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 집합에서 제외할 컬렉션.

&collection[]

[in] 현재 집합에서 제외할 컬렉션.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

IntersectWith

현재 컬렉션과 통과된 컬렉션(배열)의 교차점 연산을 생성. 지정된 컬렉션(배열)에 있는 요소만 포함하도록 현재 컬렉션을 수정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void IntersectWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void IntersectWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 집합과 교차할 컬렉션.

&collection[]

[in] 현재 세트가 교차될 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

현재 컬렉션과 통과된 컬렉션(배열) 간의 대칭 차이를 연산을 생성. 소스 개체 또는 지정된 컬렉션(배열)에 있는 요소만 포함하도록 현재 컬렉션을 수정합니다.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void SymmetricExceptWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 대칭 차이를 생성할 컬렉션.

&collection[]

[in] 대칭적 차이를 생성하는 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

UnionWith

현재 컬렉션과 통과된 컬렉션(배열)의 조합을 생성. 지정된 컬렉션(배열)에서 누락된 요소를 현재 컬렉션(배열)에 추가.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void UnionWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void UnionWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 세트가 통합될 컬렉션.

&collection[]

[in] 현재 세트가 통합될 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

현재 집합이 지정된 컬렉션 또는 배열의 적절한 하위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsProperSubsetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

관계를 결정할 배열.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

현재 집합이 지정된 컬렉션 또는 배열의 적절한 상위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsProperSupersetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 집합이 올바른 상위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

현재 집합이 지정된 컬렉션의 하위 집합인지 또는 배열의 하위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsSubsetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 집합이 하위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

현재 집합이 지정된 컬렉션 또는 배열의 상위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsSupersetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 집합이 상위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Overlaps

현재 집합이 지정된 컬렉션 또는 배열과 겹치는 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool Overlaps(
   ICollection<T>*  collection     // 비교할 컬렉션
   );
```

배열 작업을 위한 버전.

```
bool Overlaps(
   T&  array[]                     // 비교할 수 있는 배열
   );
```

매개변수

collection

[in] 중복 여부를 결정할 컬렉션.

&collection[]

[in] 중복 여부를 결정할 배열.

값 반환

현재 집합과 컬렉션 또는 배열이 겹치면 true를 반환하고 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

SetEquals

현재 집합에 지정된 컬렉션 또는 배열의 모든 요소가 포함되어 있는지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool SetEquals(
   ICollection<T>*  collection     // 비교할 컬렉션
   );
```

배열 작업을 위한 버전.

```
bool SetEquals(
   T&  array[]                     // 비교할 수 있는 배열
   );
```

매개변수

collection

[in] 요소를 비교할 컬렉션.

&collection[]

[in] 요소를 비교할 컬렉션.

값 반환

현재 집합에 지정된 컬렉션 또는 배열의 모든 요소가 포함되어 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CDefaultComparer<T>

CDefaultComparer<T>는 글로벌 메서드 비교를 기반으로 IComparer<T> 일반 인터페이스를 구현하는 도우미 클래스입니다.

설명

사용자가 IComparer<T> 인터페이스를 구현하는 다른 클래스를 암시적으로 사용하지 않는 한 CDefaultComparer<T> 클래스 일반 데이터 컬렉션에서 기본적으로 사용됩니다.

선언

```
template<typename T>
class CDefaultComparer : public IComparer<T>
```

헤더

```
#include <Generic\Internal\DefaultComparer.mqh>
```

상속 계층

IComparer

CDefaultComparer

클래스 메서드

메서드	설명
Compare	T 유형의 두 값 비교하기

© 2000-2025, MetaQuotes Ltd.

Compare

T 유형의 두 값 비교하기.

```
int Compare(
   T  x,      // 첫 번째 값
   T  y       // 두 번째 값
   );
```

매개변수

x

[in] 비교할 첫 번째 값.

y

[in] 비교할 두 번째 값.

값 반환

비교된 두 값의 비율을 나타내는 숫자를 반환:

- 결과가 0보다 작으면, x가 y보다 작음 (x<y)
- 결과가 0이면, x가 y와 같음 (x=y)
- 결과가 0보다 크면, x가 y보다 큼 (x>y)

참고

x 와 y 값은 T 유형에 따라 글로벌 비교 메서드의 오버로드 중 하나의 기준으로 비교됩니다.

© 2000-2025, MetaQuotes Ltd.

CDefaultEqualityComparer<T>

CDefaultEqualityComparer<T>는 Equals<T> 및 GetHashCode 글로벌 메서드를 기반으로 IEqualityComparer<T> 일반 인터페이스를 구현하는 도우미 클래스입니다.

설명

사용자가 IEqualityComparer<T> 인터페이스를 구현하는 다른 클래스를 암시적으로 사용하지 않는 한 CDefaultEqualityComparer<T> 클래스 제네릭 데이터 컬렉션에서 기본적으로 사용됩니다.

선언

```
template<typename T>
class CDefaultEqualityComparer : public IEqualityComparer<T>
```

헤더

```
#include <Generic\Internal\DefaultEqualityComparer.mqh>
```

상속 계층

IEqualityComparer

CDefaultEqualityComparer

클래스 메서드

메서드	설명
Equals	T 유형의 두 값 비교하기
HashCode	T 유형 개체를 기준으로 해시 코드 값을 계산

© 2000-2025, MetaQuotes Ltd.

Equals

T 유형의 두 값 비교하기.

```
bool Equals(
   T  x,      // 첫 번째 값
   T  y       // 두 번째 값
   );
```

매개변수

x

[in] 비교할 첫 번째 값.

y

[in] 비교할 두 번째 값.

값 반환

값이 같으면 true를 반환하고 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

HashCode

T 유형 개체를 기준으로 해시 코드 값을 값을 계산.

```
int HashCode(
   T  value     // 계산 대상
   );
```

매개변수

값

[in] 해시 코드를 가져올 개체.

값 반환

해시 코드 반환.

© 2000-2025, MetaQuotes Ltd.

CRedBlackTreeNode<T>

CRedBlackTreeNode<T>는 CRedBlackTree<T> 클래스를 구현하는 데 사용되는 도우미 클래스입니다.

설명

CRedBlackTreeNode<T> 클래스는 CRedBlackTree<T>의 노드입니다. 트리 탐색 메서드는 클래스에 구현됩니다.

선언

```
template<typename T>
class CRedBlackTreeNode
```

헤더

```
#include <Generic\RedBlackTree.mqh>
```

클래스 메서드

메서드	설명
값	노드 값을 반환하고 설정
Parent	상위 노드에 대한 포인터를 반환하고 설정
Left	포인터를 반환하고 왼쪽 노드에 설정
Right	포인터를 반환하고 오른쪽 노드에 설정
Color	노드 색상을 반환하고 설정
IsLeaf	지정된 노드가 리프(leaf)인지 여부를 결정
CreateEmptyNode	상위 및 하위 노드가 없는 새 블랙 노드를 만들고 포인터를 반환

© 2000-2025, MetaQuotes Ltd.

Value (the Get method)

노드 값을 반환.

```
T Value();
```

값 반환

노드 값을 반환.

Value (the Set method)

노드 값을 설정

```
void Value(
   T  value     // 노드 값
   );
```

매개변수

값

[in] 노드 값.

© 2000-2025, MetaQuotes Ltd.

Parent (the Get method)

상위 노드에 대한 포인터를 반환.

```
CRedBlackTreeNode<T>* Parent();
```

값 반환

상위 노드에 대한 포인터를 반환.

Parent (the Set method)

상위 노드에 대한 포인터를 설정.

```
void Parent(
   CRedBlackTreeNode<T>*  node     // 상위 노드에 대한 포인터
   );
```

매개변수

node

[in] 상위 노드에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Left (the Get method)

왼쪽 노드에 포인터를 반환.

```
CRedBlackTreeNode<T>* Left();
```

값 반환

왼쪽 노드에 포인터를 반환.

Left (the Set method)

포인터를 왼쪽 노드로 설정.

```
void Left(
   CRedBlackTreeNode<T>*  node     // 왼쪽 노드에 대한 포인터
   );
```

매개변수

node

[in] 왼쪽 노드에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Right (the Get method)

포인터를 오른쪽 노드에 반환.

```
CRedBlackTreeNode<T>* Right();
```

값 반환

포인터를 오른쪽 노드에 반환.

Right (the Set method)

포인터를 오른쪽 노드로 설정.

```
void Right(
   CRedBlackTreeNode<T>*  node     // 오른쪽 노드에 대한 포인터
   );
```

매개변수

node

[in] 오른쪽 노드에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Color (Get method)

노드 색상을 반환.

```
ENUM_RED_BLACK_TREE_NODE_TYPE Color();
```

값 반환

노드 색상을 반환.

Color (Set method)

노드 색상을 설정.

```
void Color(
   ENUM_RED_BLACK_TREE_NODE_TYPE  clr     // 노드 색상
   );
```

매개변수

clr

[in] 노드 색상.

참고

노드 색상은 ENUM_RED_BLACK_TREE_NODE_TYPE의 값을 사용하여 설정됩니다. 두 유형일 수 있습니다:

- RED_BLACK_TREE_NODE_RED — 노드의 레드 색상;
- RED_BLACK_TREE_NODE_BLACK — 노드의 블랙 색상.

© 2000-2025, MetaQuotes Ltd.

IsLeaf

지정한 노드가 리프인지 여부를 결정.

```
bool IsLeaf();
```

값 반환

노드가 리프인 경우 true를, 그렇지 않은 경우 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CreateEmptyNode

상위 및 하위 노드가 없는 새 블랙 노드를 생성하고 포인터를 반환.

```
static CRedBlackTreeNode<T>* CreateEmptyNode();
```

값 반환

새 노드에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

CLinkedListNode<T>

CLinkedListNode<T>は CLinkedListNode<T> クラスを実装するために使用されるヘルパークラスです。

説明

CLinkedListNode<T> クラスは、二重リンクリスト CLinkedListNode<T> のノードです。リスト検索メソッドはクラスに実装されています。

宣言

```
template<typename T>
class CLinkedListNode
```

ヘッダー

```
#include <Generic\LinkedList.mqh>
```

クラスメソッド

メソッド	説明
List	CLinkedList<T>へのポインターを返して設定します
Next	ポインターを返して次のノードに設定します
Previous	前のノードへのポインターを返して設定します
値	ノード値を返して設定します

© 2000-2025, MetaQuotes Ltd.

List (the Get method)

CLinkedList<T>에 대한 포인터 반환.

```
CLinkedList<T>* List();
```

값 반환

다음을 반환: CLinkedList<T> 링크된 목록에 대한 포인터.

List (the Set method)

CLinkedList<T>에 대한 포인터를 설정.

```
void List(
   CLinkedList<T>*  value     // 목록에 대한 포인터
   );
```

매개변수

value

[in] 연결된 목록 CLinkedList<T>에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Next (the Get method)

포인터를 다음 노드에 반환.

```
CLinkedListNode<T>* Next();
```

값 반환

포인터를 다음 노드에 반환.

Next (the Set method)

포인터를 다음 노드로 설정

```
void Next(
   CLinkedListNode<T>*  value     // 다음 점을 가리키는 포인터
   );
```

매개변수

value

[in] 다음 노드에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Previous (the Get method)

이전 노드에 대한 포인터를 반환.

```
CLinkedListNode<T>* Previous();
```

값 반환

이전 노드에 대한 포인터를 반환.

Previous (the Set method)

포인터를 이전 노드에 설정.

```
void Previous(
   CLinkedListNode<T>*  value     // 이전 노드에 대한 포인터
   );
```

매개변수

value

[in] 이전 노드에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Value (the Get method)

노드 값을 반환.

```
T Value();
```

값 반환

노드 값을 반환.

Value (the Set method)

노드 값을 설정

```
void Value(
   T  value     // 노드 값
   );
```

매개변수

값

[in] 노드 값.

© 2000-2025, MetaQuotes Ltd.

CKeyValuePair<TKey,TValue>

CKeyValuePair<TKey,TValue> 클래스는 키/값 페어를 구현합니다.

설명

CKeyValuePair<TKey,TValue> 클래스는 키와 키/값 페어의 값을 사용하는 방법을 구현.

선언

```
template<typename TKey, typename TValue>
class CKeyValuePair : public IComparable<CKeyValuePair<TKey,TValue>*>
```

헤더

```
#include <Generic\HashMap.mqh>
```

상속 계층

IEqualityComparable
IComparable
CKeyValuePair

클래스 메서드

메서드	설명
키	키/값 쌍의 키를 가져오고 설정
값	키/값 페어의 값을 가져오고 설정
Clone	키와 값이 현재 키와 같은 새 키/값 페어를 생성
Compare	현재 키/값 페어를 지정된 키/값 페어와 비교하기
Equals	현재 키/값 페어와 지정된 키 페어가 동일한지 확인
HashCode	키/값 페어를 기준으로 해시 값을 계산

© 2000-2025, MetaQuotes Ltd.

Key (the Get method)

키/값 페어의 키를 가져오기.

```
TKey Key();
```

값 반환

키 반환.

Key (the Set method)

키/값 페어의 키를 설정.

```
void Key(
   TKey  key     // 키
   );
```

매개변수

키

[in] 키.

© 2000-2025, MetaQuotes Ltd.

Value (the Get method)

키/값 쌍의 값을 가져오기.

```
TValue Value();
```

값 반환

값을 반환.

Value (the Set method)

키/값 쌍의 값을 설정.

```
void Value(
   TValue  value     // 값
   );
```

매개변수

값

[in] 값

© 2000-2025, MetaQuotes Ltd.

Clone

키와 값이 현재 키와 같은 새 키/값 페어를 생성.

```
TValue>* Clone();
```

값 반환

새 키/값 페어를 반환

© 2000-2025, MetaQuotes Ltd.

Compare

현재 키/값 페어를 지정된 키/값 페어와 비교.

```
int Compare(
   CKeyValuePair<TKeyTValue>*  pair     // 비교할 페어
   );
```

매개변수

pair

[in] 비교할 페어.

값 반환

현재 및 전달된 키-값 페어의 비율을 나타내는 숫자를 반환:

- 결과가 0보다 작으면 현재 키/값 페어가 전달된 키/값 페어보다 작습니다
- 결과가 0이면 현재 키/값 쌍이 전달된 키/값이 같습니다
- 결과가 0보다 크면 현재 키/값 페어가 전달된 키/값 페어보다 큽니다

참고

키/값 페어는 키를 기준으로 비교됩니다.

© 2000-2025, MetaQuotes Ltd.

Equals

현재 키/값 페어와 지정한 키/값 페어가 동일한지 확인.

```
bool Equals(
   CKeyValuePair<TKeyTValue>*  pair     // 비교할 페어
   );
```

매개변수

pair

[in] 비교할 페어

값 반환

키/값 페어가 같으면 true를, false를 반환

참고

키/값 페어는 키를 기준으로 비교됩니다.

© 2000-2025, MetaQuotes Ltd.

HashCode

키/값 페어를 기준으로 해시 값을 계산.

```
int HashCode();
```

값 반환

해시 코드 반환.

참고

키/값 ㅍ어의 해시 코드가 키 해시 코드와 같습니다.

© 2000-2025, MetaQuotes Ltd.

CArrayList<T>

CArrayList<T>는 IList<T> 인터페이스를 구현하는 일반 클래스.

설명

CArrayList<T> 클래스는 T 유형의 동적 데이터 리스트를 구현한 것입니다. 이 클래스는 인덱스별로 요소에 액세스하고 요소를 검색 및 삭제, 정렬 및 기타 작업을 수행하는 기본 방법을 제공합니다.

선언

```
template<typename T>
class CArrayList : public IList<T>
```

헤더

```
#include <Generic\ArrayList.mqh>
```

상속 계층

ICollection
  IList
    CArrayList

클래스 메서드

메서드	설명
용량	목록의 현재 용량을 가져오고 설정하기
Count	목록의 요소 수를 반환
Contains	목록에 지정된 값을 가진 요소가 포함되어 있는지 여부를 결정
TrimExcess	목록의 용량을 실제 요소 수로 설정하기
TryGetValue	지정한 인덱스에서 목록 요소를 가져오기
TrySetValue	지정한 인덱스에서 목록 요소의 값을 설정하기
Add	목록에 요소를 추가하기
AddRange	목록에 요소 모음 또는 배열을 추가하기
Insert	지정된 인덱스의 목록에 요소를 삽입하기
InsertRange	지정된 인덱스의 목록에 컬렉션 또는 요소 배열을 삽입하기
CopyTo	목록의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사하기
BinarySearch	오름차순 정렬 목록에서 지정된 값을 검색하기
IndexOf	목록에서 값이 처음 나타나는 경우 검색하기
LastIndexOf	목록에서 값이 마지막으로 발생했는지 검색하기

© 2000-2025, MetaQuotes Ltd.

メソド	説明
Clear	컬렉션에서 모든 요소를 제거
Remove	지정한 요소의 첫 번째 항목을 목록에서 제거
RemoveAt	목록의 지정된 인덱스에서 요소를 제거
RemoveRange	목록에서 요소 범위를 제거
Reverse	목록의 요소 순서를 반전
Sort	목록에서 요소 정렬

© 2000-2025, MetaQuotes Ltd.

Capacity (the Get method)

현재 목록 용량을 반환.

```
int Capacity();
```

값 반환

현재 목록 용량을 반환.

용량 (Set 메서드)

목록의 현재 용량을 설정하기.

```
void Capacity(
   const int  capacity     // 용량 값
   );
```

매개변수

용량

[in] 용량의 새 값.

© 2000-2025, MetaQuotes Ltd.

Count

목록의 요소 수를 반환.

```
int Count();
```

값 반환

요소 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

목록에 지정된 값을 가진 요소가 포함되어 있는지 여부를 결정.

```
bool Contains(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

지정한 값을 가진 요소가 목록에 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

목록의 용량을 실제 요소 수로 설정하여 사용하지 않는 메모리를 확보합니다.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

TryGetValue

지정된 인덱스에서 목록의 요소를 가져오기.

```
bool TryGetValue(
   const int  index,      // 인덱스
   T&         value       // 쓸 변수
   );
```

매개변수

인덱스

[in] 가져오려는 값 목록 요소의 인덱스.

&value

[out] 요소 값을 쓸 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

지정한 인덱스에서 목록 요소의 값을 설정.

```
bool TrySetValue(
   const int  index,      // 인덱스
   T          value       // 요소 값
   );
```

매개변수

인덱스

[in] 설정할 값을 나타내는 목록 요소의 인덱스.

값

[in] 목록 요소의 값을 설정.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Add

목록에 요소 추가하기

```
bool Add(
   T  value     // 요소의 값
   );
```

매개변수

값

[in] 추가할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

AddRange

목록에 요소 모음 또는 배열을 추가하기

배열을 추가하는 버전.

```
bool AddRange(
   const T&  array[]                // 추가할 배열
   );
```

모음을 추가하는 버전.

```
bool AddRange(
   ICollection<T>*  collection      // 추가될 모음
   );
```

매개변수

&array[]

[in] 추가될 배열.

**collection*

[in] 추가될 모음.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Insert

지정된 인덱스의 목록에 요소를 삽입하기.

```
bool Insert(
   const int  index,      // 삽입할 인덱스
   T          item        // 삽입될 값
   );
```

매개변수

인덱스

[in] 삽입할 인덱스.

항목

[in] 지정된 인덱스에 삽입될 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

InsertRange

지정한 인덱스 목록에 컬렉션 또는 요소 배열을 삽입하기.

배열을 삽입하는 버전.

```
bool InsertRange(
   const int  index,               // 삽입할 인덱스
   const T&   array[]              // 삽입될 배열
   );
```

컬렉션을 삽입하는 버전.

```
bool InsertRange(
   const int        index,         // 삽입할 인덱스
   ICollection<T>*  collection     // 삽입될 컬렉션
   );
```

매개변수

인덱스

[in] 삽입할 인덱스.

&array[]

[in] 지정된 인덱스에 삽입될 배열.

**collection*

[in] 지정된 인덱스에 삽입될 컬렉션.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CopyTo

목록의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사하기.

```
int CopyTo(
   T&          dst_array[],     // 표기 배열
   const int   dst_start=0      // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 목록의 요소를 쓸 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 요소 수를 반환

© 2000-2025, MetaQuotes Ltd.

BinarySearch

오름차순 정렬 목록에서 지정된 값을 검색하기.

요소를 비교하기 위해 IComparable<T> 인터페이스를 구현하는 클래스를 사용하여 지정된 범위잉 값을 검색하는 버전.

```
int BinarySearch(
   const int       index,        // 시작 인덱스
   const int       count,        // 검색 범위
   T               item,         // 검색 값
   IComparer<T>*   comparer      // 비교할 인터페이스
   );
```

요소를 비교하기 위해 IComparable<T> 인터페이스를 구현하는 클래스르 사용하여 검색하는 버전.

```
int BinarySearch(
   T               item,         // 검색 값
   IComparer<T>*   comparer      // 비교할 인터페이스
   );
```

요소를 비교하기 위해 ::Compare 글로벌 메서드를 사용하여 검색하는 버전.

```
int BinarySearch(
   T  item                       // 검색 값
   );
```

매개변수

인덱스

[in] 검색이 시작되는 시작 인덱스.

카운트

[in] 검색 범위의 길이.

항목

[in] 검색된 값.

**comparer*

[in] 요소 비교를 위한 인터페이스.

값 반환

찾은 요소의 인덱스를 반환. 검색 값을 찾을 수 없으면 가장 가까운 가장 작은 요소의 인덱스를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

IndexOf

목록에서 값이 처음 나타나는 경우를 검색.

전체 목록에서 검색하는 버전.

```
int IndexOf(
   T  item                    // 검색 값
   );
```

지정한 포지션에서 목록 끝까지 검색하는 버전.

```
int IndexOf(
   T           item,          // 검색 값
   const int   start_index    // 시작 인덱스
   );
```

지정된 범위의 지정된 포지션에서 검색하는 버전.

```
int IndexOf(
   T           item,          // 검색 값
   const int   start_index,   // 시작 인덱스
   const int   count          // 검색 범위
   );
```

매개변수

항목

[in] 검색된 값.

start_index

[in] 검색이 시작되는 시작 인덱스.

카운트

[in] 검색 범위의 길이.

값 반환

처음 찾은 요소의 인덱스를 반환. 값을 찾을 수 없으면 -1을 반환.

© 2000-2025, MetaQuotes Ltd.

LastIndexOf

목록에서 값이 마지막으로 표시되는 항목을 검색.

전체 목록에서 검색하는 버전.

```
int LastIndexOf(
   T  item                    // 검색 값
   );
```

지정한 포지션에서 목록 끝까지 검색하는 버전.

```
int LastIndexOf(
   T           item,          // 검색 값
   const int   start_index    // 시작 인덱스
   );
```

지정된 범위의 지정된 포지션에서 검색하는 버전.

```
int LastIndexOf(
   T           item,          // 검색 값
   const int   start_index,   // 시작 인덱스
   const int   count          // 검색 범위
   );
```

매개변수

항목

[in] 검색된 값.

start_index

[in] 검색이 시작되는 시작 인덱스.

카운트

[in] 검색 범위의 길이.

값 반환

마지막으로 찾은 요소의 인덱스를 반환. 값을 찾을 수 없으면 -1을 반환.

© 2000-2025, MetaQuotes Ltd.

Clear

모음의 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

지정한 요소의 첫 번째 항목을 목록에서 제거.

```
bool Remove(
   T  item     // 요소 값
   );
```

매개변수

항목

[in] 삭제할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

RemoveAt

목록의 지정된 인덱스에서 요소를 제거.

```
bool RemoveAt(
   const int  index     // 인덱스
   );
```

매개변수

인덱스

[in] 제거할 요소의 인덱스.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

RemoveRange

목록에서 요소 범위를 제거.

```
bool RemoveRange(
   const int  start_index,     // 시작 인덱스
   const int  count            // 요소수
   );
```

매개변수

start_index

[in] 삭제가 시작되는 시작 인덱스.

카운트

[in] 삭제할 요소 수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Reverse

목록에서 요소 순서를 반전.

전체 목록으로 작업하기 위한 버전.

```
bool Reverse();
```

지정된 목록 요소 범위로 작업하기 위한 버전.

```
bool Reverse(
   const int  start_index,     // 시작 인덱스
   const int  count            // 요소 수
   );
```

매개변수

start_index

[in] 시작 인덱스.

카운트

[in] 작업에 참여하는 목록 요소의 수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Sort

목록에서 요소를 정렬.

목록의 모든 요소를 정렬하는 버전.

```
bool Sort();
```

요소를 비교하기 위해 IComparable<T> 인터페이스를 구현하는 클래스를 사용하여 목록의 모든 요소를 정렬하는 버전.

```
bool Sort(
   IComparer<T>*  comparer          // 비교할 인터페이스
   );
```

요소를 비교하기 위해 IComparable<T> 인터페이스를 구현하는 클래스를 사용하여 목록에 지정된 범위의 요소를 정렬하는 버전.

```
bool Sort(
   const int  start_index,      // 시작 인덱스
   const int  count             // 요소 수
   IComparer<T>*  comparer      // 비교할 인터페이스
   );
```

매개변수

comparer

[in] 요소 비교를 위한 인터페이스.

start_index

[in] 정렬이 시작되는 시작 인덱스.

카운트

[in] 정렬 범위의 길이.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CHashMap<TKey, TValue>

CHashMap<TKey, TValue>는 IMap<TKey, TValue> 인터페이스를 구현하는 일반 클래스입니다.

설명

CHashMap<TKey, TValue> 클래스는 동적 해시 테이블의 구현으로, 데이터는 주요 고유성 요구사항을 고려하여 정렬되지 않은 키/값 페어의 형태로 저장됩니다. 이 클래스는 키별 값에 액세스하고 키/값 페어를 검색 및 삭제하는 등의 해시 테이블 관련 작업을 위한 기본 메서드를 제공합니다.

선언

```
template<typename TKey, typename TValue>
class CHashMap : public IMap<TKey, TValue>
```

헤더

```
#include <Generic\HashMap.mqh>
```

상속 계층

ICollection
  IMap
    CHashMap

클래스 메서드

메서드	설명
Add	해시 테이블에 키/값 페어를 추가
Count	해시 테이블의 요소 수를 반환
Comparer	해시 테이블을 구성하는 데 사용되는 IEqualityComparer<T> 인터페이스에 대한 포인터를 반환
Contains	해시 테이블에 지정된 키/값 페어가 포함되어 있는지 여부를 결정
ContainsKey	해시 테이블에 지정된 키와 키/값 페어가 포함되어 있는지 여부를 결정
ContainsValue	CHashMap<TKey, TValue>는 IMap<TKey, TValue> 인터페이스를 구현하는 일반 클래스입니다
CopyTo	지정된 인덱스에서 시작하여 해시 테이블의 모든 키/값 페어를 지정된 배열로 복사
Clear	해시 테이블에서 모든 요소를 제거
Remove	해시 테이블에서 키/값 페어의 첫 번째 항목을 제거
TryGetValue	해시 테이블에서 지정된 키를 가진 요소 가져오기
TrySetValue	지정된 키의 해시 테이블에서 키/값 페어를 변경

© 2000-2025, MetaQuotes Ltd.

Add

해시 테이블에 키/값 페어를 추가.

생성된 키/값 페어를 추가하는 버전.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // 키/값 페어
   );
```

지정된 키 및 값을 사용하여 새 키/값 페어를 추가하는 버전.

```
bool Add(
   TKey    key,                         // 키
   TValue  value                        // 값
   );
```

매개변수

pair

[in] 키/값 페어.

키

[in] 키.

값

[in] 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

해시 테이블의 요소 수를 반환.

```
int Count();
```

© 2000-2025, MetaQuotes Ltd.

Comparer

해시 테이블을 구성하는 데 사용되는 IEqualityComparer<T> 인터페이스에 대한 포인터를 반환.

```
IEqualityComparer<TKey>* Comparer() const;
```

값 반환

IEqualityComparer<T> 인터페이스에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

다음을 결정: 해시 테이블에 지정된 키/값 페어가 포함되어 있는지 여부.

생성된 키/값 페어로 작업하기 위한 버전.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // 키/값 페어
   );
```

키/값 페어로 별도 설정된 키와 값의 형태로 작업하기 위한 버전.

```
bool Contains(
   TKey   key,                          // 키
   TValue value                         // 값
   );
```

매개변수

item

[in] 키/값 페어.

키

[in] 키.

값

[in] 값.

값 반환

해시 테이블에 지정된 키 및 값을 가진 키/값 페어가 포함되어 있으면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

ContainsKey

다음을 결정:해시 테이블에 지정된 키와 키/값 페어가 포함되어 있는지 여부.

```
bool ContainsKey(
   TKey  key     // 키
   );
```

매개변수

키

[in] 키.

값 반환

해시 테이블에 지정된 키를 가진 키/값 페어가 포함되어 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

ContainsValue

다음을 결정: 해시 테이블에 지정된 값의 키/값 페어가 포함되어 있는지 여부.

```
bool ContainsValue(
   TValue  value     // 값
   );
```

매개변수

값

[in] 값.

값 반환

해시 테이블에 지정된 값의 키/값 페어가 포함되어 있으면 true를 반환하고, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CopyTo

지정된 인덱스에서 시작하여 해시 테이블의 모든 키/값 페어를 지정된 배열로 복사.

키/값 페어 배열에 해시 테이블을 복사하는 버전.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // 키/값 페어 표기 배열
   const int                    dst_start=0      // 표기 시작 인덱스
   );
```

해시 테이블을 키 및 값의 개별 배열에 복사하는 버전.

```
int CopyTo(
   TKey&       dst_keys[],                       // 키 표기 배열
   TValue&     dst_values[],                     // 값 표기 배열
   const int   dst_start=0                       // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 해시 테이블의 모든 페어를 기록할 배열.

&dst_keys[]

[out] 해시 테이블의 모든 키를 기록할 배열.

&dst_values[]

[out] 해시 테이블의 모든 값이 기록될 배열.

dst_start=0

[in] 복사가 시작되는 배열 인덱스.

값 반환

복사된 키/값 페어의 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Clear

해시 테이블에서 모든 요소를 제거하기.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

해시 테이블에서 키/값 페어의 첫 번째 항목을 제거.

생성된 키/값 페어를 기반으로 키-값 페어를 제거하는 버전.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // 키/값 페어"
   );
```

키를 기반으로 키-값 페어를 제거하는 버전.

```
bool Remove(
   TKey  key                            // 키
   );
```

매개변수

item

[in] 키/값 페어.

키

[in] 키.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

해시 테이블에서 지정한 키를 가진 요소를 가져오기.

```
bool TryGetValue(
   TKey      key,       // 키
   TValue&   value      // 값 표기 변수
   );
```

매개변수

키

[in] 키.

&value

[out] 키/값 페어의 지정된 값을 쓸 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

지정된 키의 해시 테이블에서 키/값 페어 값을 변경.

```
bool TrySetValue(
   TKey    key,       // 키
   TValue  value      // 새 값
   );
```

매개변수

키

[in] 키.

값

[in] 지정된 키-값 페어에 할당할 새 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CHashSet<T>

CHashSet<T>는 ISet<T> 인터페이스를 구현하는 일반 클래스.

설명

CHashSet<T> 클래스는 각 값의 필요한 고유성을 가진 T 유형의 순서가 없는 동적 데이터 집합의 구현입니다. 이 클래스는 집합의 결합 및 교차점, 엄격하지 않은 하위 집합의 정의 등과 같은 집합 및 관련 연산을 사용하여 기본 메서드를 제공합니다.

선언

```
template<typename T>
class CHashSet : public ISet<T>
```

헤더

```
#include <Generic\HashSet.mqh>
```

상속 계층

ICollection
  ISet
    CHashSet

클래스 메서드

메서드	설명
Add	집합에 요소를 추가
Count	집합의 요소 수를 반환
Comparer	집합에 지정된 값의 요소가 포함되어 있는지 여부를 결정
Contains	집합을 구성하는 데 IEqualityComparer<T> 인터페이스에 대한 포인터를 반환
TrimExcess	세트의 용량을 실제 요소 수로 설정하여 사용하지 않는 메모리를 확보
CopyTo	집합의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사
Clear	집합에서 모든 요소를 제거
Remove	집합에서 지정한 요소를 제거
ExceptWith	현재 컬렉션과 통과된 컬렉션(배열) 간의 차이 연산을 생성
IntersectWith	현재 컬렉션과 통과된 컬렉션(배열)의 교차점 연산을 생성
SymmetricExceptWith	현재 컬렉션과 통과된 컬렉션(배열) 간의 대칭 차이 연산을 생성
UnionWith	현재 컬렉션과 통과된 컬렉션(배열)의 결합을 생성
IsProperSubsetOf	현재 집합이 지정된 컬렉션 또는 배열의 적절한 부분 집합인지 여부를 결정

© 2000-2025, MetaQuotes Ltd.

メソッド	説明
IsProperSupersetOf	現在 集合が 指定された コレクション または 配列の 適切な 上位 集合かどうか 決定
IsSubsetOf	現在 集合が 指定された コレクション または 配列の 下位 集合かどうか 決定
IsSupersetOf	現在 集合が 指定された コレクション または 配列の 上位 集合かどうか 決定
Overlaps	現在 集合が 指定された コレクション または 配列と 交差するか どうか 決定
SetEquals	現在 集合に 指定された コレクション または 配列の すべて 要素が 含まれて いるか どうか 決定

© 2000-2025, MetaQuotes Ltd.

Add

집합에 요소를 추가.

```
bool Add(
   T  value     // 요소의 값
   );
```

매개변수

값

[in] 추가할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

집합의 요소 수를 반환.

```
int Count();
```

값 반환

요소 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

집합에 지정된 값의 요소가 포함되어 있는지 여부를 결정.

```
bool Contains(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

지정한 값을 가진 요소가 집합에 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Comparer

집합을 구성하는 데 사용되는 IEqualityComparer<T> 인터페이스에 대한 포인터를 반환.

```
IEqualityComparer<T>* Comparer() const;
```

값 반환

IEqualityComparer<T> 인터페이스에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

세트의 용량을 실제 요소 수로 설정하여 사용하지 않는 메모리를 확보.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

集合의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사.

```
int CopyTo(
   T&          dst_array[],     // 표기 위한 배열
   const int   dst_start=0      // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 집합의 요소를 쓸 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 요소 수를 반환

© 2000-2025, MetaQuotes Ltd.

Clear

집합에서 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

집합에서 지정한 요소를 제거.

```
bool Remove(
   T  item     // 요소 값
   );
```

매개변수

항목

[in] 삭제할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

ExceptWith

현재 컬렉션과 통과된 컬렉션(배열) 간의 차이 연산을 생성. 지정된 컬렉션(배열)에 있는 모든 요소를 현재 컬렉션(배열)에서 제거.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void ExceptWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void ExceptWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 집합에서 제외할 컬렉션.

&collection[]

[in] 현재 집합에서 제외할 컬렉션.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

IntersectWith

현재 컬렉션과 통과된 컬렉션(배열)의 교차점 연산을 생성. 지정된 컬렉션(배열)에 있는 요소만 포함하도록 현재 컬렉션을 수정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void IntersectWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void IntersectWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 집합과 교차할 컬렉션.

&collection[]

[in] 현재 세트가 교차될 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

현재 컬렉션과 통과된 컬렉션(배열) 간의 대칭 차이를 연산을 생성. 소스 개체 또는 지정된 컬렉션(배열)에 있는 요소만 포함하도록 현재 컬렉션을 수정합니다.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void SymmetricExceptWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 대칭 차이를 생성할 컬렉션.

&collection[]

[in] 대칭적 차이를 생성하는 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

UnionWith

현재 컬렉션과 통과된 컬렉션(배열)의 조합을 생성. 지정된 컬렉션(배열)에서 누락된 요소를 현재 컬렉션(배열)에 추가.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void UnionWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void UnionWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 세트가 통합될 컬렉션.

&collection[]

[in] 현재 세트가 통합될 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

현재 집합이 지정된 컬렉션 또는 배열의 적절한 하위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsProperSubsetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

관계를 결정할 배열.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

현재 집합이 지정된 컬렉션 또는 배열의 적절한 상위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsProperSupersetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 집합이 올바른 상위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

현재 집합이 지정된 컬렉션의 하위 집합인지 또는 배열의 하위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsSubsetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 집합이 하위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

현재 집합이 지정된 컬렉션 또는 배열의 상위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsSupersetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 집합이 상위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Overlaps

현재 집합이 지정된 컬렉션 또는 배열과 겹치는 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool Overlaps(
   ICollection<T>*  collection     // 비교할 컬렉션
   );
```

배열 작업을 위한 버전.

```
bool Overlaps(
   T&  array[]                     // 비교할 수 있는 배열
   );
```

매개변수

collection

[in] 중복 여부를 결정할 컬렉션.

&collection[]

[in] 중복 여부를 결정할 배열.

값 반환

현재 집합과 컬렉션 또는 배열이 겹치면 true를 반환하고 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

SetEquals

현재 집합에 지정된 컬렉션 또는 배열의 모든 요소가 포함되어 있는지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool SetEquals(
   ICollection<T>*  collection     // 비교할 컬렉션
   );
```

배열 작업을 위한 버전.

```
bool SetEquals(
   T&  array[]                     // 비교할 수 있는 배열
   );
```

매개변수

collection

[in] 요소를 비교할 컬렉션.

&collection[]

[in] 요소를 비교할 배열.

값 반환

현재 집합에 지정된 컬렉션 또는 배열의 모든 요소가 포함되어 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CLinkedList<T>

CLinkedList<T>는 ICollection<T> 인터페이스를 구현하는 일반 클래스 interface.

설명

CLinkedList<T> 클래스는 T 유형의 동적 이중 링크 데이터 목록을 구현한 것입니다. 이 클래스는 추가, 삭제, 검색 요소 등과 같이 이중으로 연결된 목록에서 작업할 수 있는 기본 방법을 제공합니다.

선언

```
template<typename T>
class CLinkedList : public ICollection<T>
```

헤더

```
#include <Generic\LinkedList.mqh>
```

상속 계층

ICollection

CLinkedList

클래스 메서드

메서드	설명
Add	링크된 목록에 요소를 추가
AddAfter	링크된 목록에서 지정된 노드 뒤에 요소를 추가
AddBefore	링크된 목록에서 지정된 노드 앞에 요소를 추가
AddFirst	링크된 목록의 시작 부분에 요소를 추가
AddLast	연결된 목록의 끝에 요소를 추가
Count	링크된 목록의 요소 수를 반환
Head	링크된 목록의 첫 번째 노드에 대한 포인터를 반환
First	링크된 목록의 첫 번째 노드에 대한 포인터를 반환
Last	링크된 목록의 노드에 대한 포인터를 반환
Contains	링크된 목록에 지정된 값을 가진 요소가 포함되어 있는지 여부를 결정
CopyTo	링크된 목록의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사
Clear	연결된 목록에서 모든 요소를 제거
Remove	링크된 목록에서 지정된 요소의 첫 번째 항목을 제거
RemoveFirst	링크된 목록의 첫 번째 요소를 제거
RemoveLast	연결된 목록의 마지막 요소를 제거

© 2000-2025, MetaQuotes Ltd.

メソド	説明
Find	リンクされた目録において指定された値の最初の発生項目検索
FindLast	リンクされた目録において指定された値の最後の項目を検索

© 2000-2025, MetaQuotes Ltd.

Add

연결된 목록에 요소를 추가.

```
bool Add(
   T  value     // 요소의 값
   );
```

매개변수

값

[in] 추가할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

AddAfter

링크된 목록에서 지정된 노드 뒤에 요소를 추가.

값을 기준으로 요소를 추가하는 버전.

```
CLinkedListNode<T>* AddAfter(
   CLinkedListNode<T>*  node,         // 요소를 추가할 노드
   T                    value         // 추가할 요소
   );
```

값 반환

추가된 노드에 대한 포인터를 반환.

값을 기준으로 요소를 형성된 노드로 추가하는 버전.

```
bool AddAfter(
   CLinkedListNode<T>*  node,         // 요소를 추가할 노드
   CLinkedListNode<T>*  new_node      // 추가될 노드
   );
```

매개변수

node

[in] 링크된 목록의 노드이며, 이 노드 뒤에 새 요소에 추가.

값

[in] 추가될 요소..

new_node

[in] 추가될 노드.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

AddBefore

링크된 목록에서 지정된 노드 앞에 요소를 추가.

값을 기준으로 요소를 추가하는 버전.

```
CLinkedListNode<T>* AddBefore(
   CLinkedListNode<T>*  node,         // 요소를 추가할 노드
   T                    value         // 추가할 요소
   );
```

값 반환

추가된 노드에 대한 포인터를 반환.

값을 기준으로 요소를 형성된 노드로 추가하는 버전.

```
bool AddBefore(
   CLinkedListNode<T>*  node,         // 요소를 추가할 노드
   CLinkedListNode<T>*  new_node      // 추가될 노드
   );
```

매개변수

node

[in] 새 요소를 추가하기 전에 연결된 목록의 노드.

값

[in] 추가될 요소..

new_node

[in] 추가될 노드.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

AddFirst

링크된 목록의 시작 부분에 요소를 추가.

값을 기준으로 요소를 추가하는 버전.

```
CLinkedListNode<T>* AddFirst(
   T    value                    // 추가할 요소
   );
```

값 반환

추가된 노드에 대한 포인터를 반환.

값을 기준으로 요소를 형성된 노드로 추가하는 버전.

```
bool AddFirst(
   CLinkedListNode<T>*  node     // 츄가할 노드
   );
```

매개변수

값

[in] 추가될 요소..

**node*

[in] 추가될 노드.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

AddLast

연결된 목록의 끝에 요소를 추가

값을 기준으로 요소를 추가하는 버전.

```
CLinkedListNode<T>* AddLast(
   T  value                        // 추가할 요소
   );
```

값 반환

추가된 노드에 대한 포인터를 반환.

값을 기준으로 요소를 형성된 노드로 추가하는 버전.

```
bool AddLast(
   CLinkedListNode<T>*  node      // 추가할 노드
   );
```

매개변수

값

[in] 추가될 요소..

**node*

[in] 추가될 노드.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

링크된 목록의 요소 수를 반환.

```
int Count();
```

값 반환

요소 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Head

연결된 목록의 첫 번재 노드에 대한 포인터를 반환.

```
CLinkedListNode<T>* Head();
```

값 반환

첫 번째 노드에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

First

연결된 목록의 첫 번재 노드에 대한 포인터를 반환.

```
CLinkedListNode<T>* First();
```

값 반환

첫 번째 노드에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

Last

연결된 목록의 마지막 노드에 대한 포인터를 반환.

```
CLinkedListNode<T>* Last();
```

값 반환

마지막 노드에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

링크된 목록에 지정된 값을 가진 요소가 포함되어 있는지 여부를 결정.

```
bool Contains(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

지정된 값을 가진 요소가 링크된 목록에 있으면 true를, 그렇지 않으면 false를 반화.

© 2000-2025, MetaQuotes Ltd.

CopyTo

링크된 목록의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사.

```
int CopyTo(
   T&          dst_array[],     // 표기 배열
   const int   dst_start=0      // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 링크된 목록의 요소를 쓸 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 요소 수를 반환

© 2000-2025, MetaQuotes Ltd.

Clear

모음의 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

링크된 목록에서 지정된 요소의 첫 번째 항목을 제거.

값을 기준으로 요소를 제거하는 버전.

```
bool Remove(
   T  item                        // 요소 값
   );
```

노드에 대한 포인터로 요소를 제거하는 버전.

```
bool Remove(
   CLinkedListNode<T>*  node      // 요소 노드
   );
```

매개변수

항목

[in] 삭제할 요소의 값.

**node*

[in] 삭제할 요소의 노드.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

RemoveFirst

링크된 목록의 첫 번째 요소를 제거.

```
bool RemoveFirst();
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

RemoveLast

링크된 목록의 마지막 요소를 제거.

```
bool RemoveLast();
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Find

링크된 목록에서 지정된 값이 처음 나타나는 경우를 검색.

```
CLinkedListNode<T>* Find(
   T  value     // 검색 값
   );
```

매개변수

값

[in] 검색된 값.

값 반환

성공 시 검색 값을 포함하는 첫 번째 발견된 노드에 대한 포인터를 반환하거나 NULL을 반환.

© 2000-2025, MetaQuotes Ltd.

FindLast

링크된 목록에서 지정된 값의 마지막 항목을 검색.

```
CLinkedListNode<T>* FindLast(
   T  value     // 검색 값
   );
```

매개변수

값

[in] 검색된 값.

값 반환

성공 시 검색 값을 포함하는 마지막으로 찾은 노드에 대한 포인터를 반환하거나 그렇지 않으면 NULL을 반환.

© 2000-2025, MetaQuotes Ltd.

CQueue<T>

CQueue<T>는 ICollection<T> 인터페이스를 구현하는 일반 클래스입니다.

설명

CQueue<T> 클래스는 FIFO (선입선출) 원칙에 따라 작동하는 대기열로 구성된 T 유형 데이터의 동적 컬렉션입니다.

선언

```
template<typename T>
class CQueue : public ICollection<T>
```

헤더

```
#include <Generic\Queue.mqh>
```

상속 계층

ICollection

CQueue

클래스 메서드

메서드	설명
Add	요소를 대기열에 추가
Enqueue	요소를 대기열에 추가
Count	대기열에 있는 요소 수를 반환
Contains	대기열에 지정된 값의 요소가 포함되어 있는지 여부를 결정
TrimExcess	대기열의 용량을 실제 요소 수로 설정하여 사용하지 않는 메모리를 확보
CopyTo	대기열의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사
Clear	대기열에서 모든 요소를 제거
Remove	대기열에서 지정한 요소의 첫 번째 항목을 제거
Dequeue	시작 요소를 반환하고 대기열에서 제거
Peek	대기열에서 제거하지 않고 시작 요소를 반환

© 2000-2025, MetaQuotes Ltd.

Add

요소를 대기열에 추가.

```
bool Add(
   T  value     // 요소의 값
   );
```

매개변수

값

[in] 추가할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Enqueue

요소를 대기열에 추가.

```
bool Enqueue(
   T  value     // 추가될 요소
   );
```

매개변수

값

[in] 추가될 요소..

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

대기열에 있는 요소 수를 반환.

```
int Count();
```

값 반환

요소 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

대기열에 지정된 값의 요소가 포함되어 있는지 여부를 결정.

```
bool Contains(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

지정한 값을 가진 요소가 대기열에 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

대기열의 용량을 실제 요소 수로 설정하여 사용하지 않는 메모리를 확보.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

대기열의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사.

```
int CopyTo(
   T&          dst_array[],     // 표기 배열
   const int   dst_start=0      // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 대기열의 요소가 기록될 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 요소 수를 반환

© 2000-2025, MetaQuotes Ltd.

Clear

대기열에서 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

대기열에서 지정한 요소의 첫 번째 항목을 제거.

```
bool Remove(
   T  item     // 요소 값
   );
```

매개변수

항목

[in] 삭제할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Dequeue

첫 번째 요소를 반환하고 대기열에서 제거.

```
T Dequeue();
```

값 반환

시작 요소를 반환.

© 2000-2025, MetaQuotes Ltd.

Peek

대기열에서 제거하지 않는 첫 번째 요소를 반환.

```
T Peek();
```

값 반환

시작 요소를 반환.

© 2000-2025, MetaQuotes Ltd.

CRedBlackTree<T>

CRedBlackTree<T>는 ICollection<T> 인터페이스를 구현하는 일반 클래스.

설명

CRedBlackTree<T> 클래스는 노드가 T 유형 데이터를 저장하는 동적 레드-블랙 트리의 구현입니다. 클래스는 레드-블랙 트리에서 작업하는 기본 메서드(예: 최대값 및 최소값 검색)를 제공합니다.

선언

```
template<typename T>
class CRedBlackTree : public ICollection<T>
```

헤더

```
#include <Generic\RedBlackTree.mqh>
```

상속 계층

ICollection

CRedBlackTree

클래스 메서드

메서드	설명
Add	레드-블랙 트리에 요소를 추가
Root	레드-블랙 트리의 루트에 대한 포인터를 반환
Count	레드-블랙 트리의 요소 수를 반환
Contains	레드-블랙 트리에 지정된 값의 요소가 포함되어 있는지 여부를 결정
Comparer	레드-블랙 트리를 구성하는 데 사용되는 IComparer<T> 인터페이스에 대한 포인터를 반환
TryGetMin	레드-블랙 트리의 최소 요소를 가져오기
TryGetMax	레드-블랙 트리의 최대 요소를 가져오기
CopyTo	레드-블랙 트리의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사
Clear	레드-블랙 트리에서 모든 요소를 제거
Remove	레드-블랙 트리에서 지정한 요소의 발생을 제거
RemoveMin	레드-블랙 트리에서 최소값의 요소를 제거
RemoveMax	레드-블랙 트리에서 최대값의 요소를 제거
Find	레드-블랙 트리에서 지정된 값의 발생을 검색
FindMax	레드-블랙 트리에서 최대값의 요소를 검색

© 2000-2025, MetaQuotes Ltd.

メソッド	説明
FindMin	レッド-ブラック トリーで最小値の要素を検索

© 2000-2025, MetaQuotes Ltd.

Add

레드-블랙 트리에 요소를 추가.

```
bool Add(
   T  value     // 추가될 요소
   );
```

매개변수

값

[in] 추가될 요소..

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

레드-블랙 트리의 요소 수를 반환.

```
int Count();
```

값 반환

요소 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Root

레드-블랙 트리의 루트에 대한 포인터를 반환.

```
CRedBlackTreeNode<T>* Root();
```

값 반환

루트에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

레드-블랙 트리에 지정된 값의 요소가 포함되어 있는지 여부를 결정.

```
bool Contains(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

지정된 값을 가진 요소가 레드-블랙 트리에서 발견되면 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Comparer

레드-블랙 트리를 구성하는 데 사용되는 IComparer<T> 인터페이스에 대한 포인터를 반환.

```
IComparer<T>* Comparer() const;
```

값 반환

IComparer<T> 인터페이스에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

TryGetMin

레드-블랙 트리의 최소 요소를 가져오기.

```
bool TryGetMin(
   T&  min     // 값 표기 변수
   );
```

매개변수

&min

[out] 최소값을 기록할 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TryGetMax

레드-블랙 트리의 최대 요소를 가져오기.

```
bool TryGetMax(
   T&  max     // 값 표기 변수
   );
```

매개변수

&max

[out] 최대값을 기록할 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CopyTo

레드-블랙 트리의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사.

```
int CopyTo(
   T&          dst_array[],     // 표기 위한 배열
   const int   dst_start=0      // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 레드-블랙 트리의 요소가 기록될 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 요소 수를 반환

© 2000-2025, MetaQuotes Ltd.

Clear

레드-블랙 트리에서 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

레드-블랙 트리에서 지정한 요소의 발생을 제거.

지정된 값을 가진 요소를 제거하는 버전.

```
bool Remove(
   T  value                         // 요소 값
   );
```

노드에 대한 포인터로 요소를 제거하는 버전.

```
bool Remove(
   CRedBlackTreeNode<T>*  node      // 요소 노드
   );
```

매개변수

항목

[in] 삭제할 요소의 값.

**node*

[in] 삭제할 요소의 노드.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

RemoveMin

레드-블랙 트리에서 최소값의 요소를 제거.

```
bool RemoveMin();
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

RemoveMax

레드-블랙 트리에서 최대값 요소를 제거.

```
bool RemoveMax();
```

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Find

레드-블랙 트리에서 지정된 값의 발생을 검색.

```
CRedBlackTreeNode<T>* Find(
   T  value     // 검색 값
   );
```

매개변수

값

[in] 검색된 값.

값 반환

검색 성공 시 검색 값을 포함하는 찾은 노드에 대한 포인터를 반환하거나 그렇지 않으면 NULL을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

FindMin

레드-블랙 트리에서 최소값의 요소를 검색.

```
CRedBlackTreeNode<T>* FindMin();
```

값 반환

성공 시 최소값이 포함된 노드에 대한 포인터를 반환하거나, 그렇지 않으면 NULL을 반환.

© 2000-2025, MetaQuotes Ltd.

FindMax

레드0블랙 트리에서 최대값 요소를 검색.

```
CRedBlackTreeNode<T>* FindMax();
```

값 반환

성공 시 최대값이 포함된 노드에 대한 포인터를 반환하거나 그렇지 않으면 NULL을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CSortedMap<TKey,TValue>

CSortedMap<TKey,TValue>는 IMap<TKey,TValue> 인터페이스를 구현하는 일반 클래스입니다.

설명

The CSortedMap<TKey,TValue> 클래스는 데이터를 키별로 정렬하고 키 고유성 요구사항을 고려하여 키/값 페어로 저장하는 동적 해시 테이블의 구현입니다. 이 클래스는 키별 값에 액세스하고 키/값 페어를 검색 및 삭제하는 등의 해시 테이블 관련 작업을 위한 기본 메서드를 제공합니다.

선언

```
template<typename TKey, typename TValue>
class CSortedMap : public IMap<TKey, TValue>
```

헤더

```
#include <Generic\SortedMap.mqh>
```

상속 계층

ICollection
  IMap
    CSortedMap

클래스 메서드

메서드	설명
Add	해시 테이블에 키/값 페어를 추가
Count	정렬된 해시 테이블의 요소 수를 반환
Contains	정렬된 해시 테이블에 지정된 키/값 테이블이 포함되어 있는지 여부를 결정
ContainsKey	정렬된 해시 테이블에 지정된 키가 있는 키/값 테이블이 포함되어 있는지 여부를 결정
ContainsValue	정렬된 해시 테이블에 지정된 값의 키/값 테이블이 포함되어 있는지 여부를 결정
Comparer	정렬된 해시 테이블을 구성하는 데 사용되는 IComparer<T> 인터페이스에 대한 포인터를 반환
CopyTo	정렬된 해시 테이블의 모든 키/값 페어를 지정된 인덱스에서 시작하여 지정된 배열로 복사
Clear	정렬된 해시 테이블에서 모든 요소를 제거
Remove	정렬된 해시 테이블에서 키/값 페어의 첫 번째 항목을 제거
TryGetValue	정렬된 해시 테이블에서 지정된 키를 가진 요소를 가져오기
TrySetValue	정렬된 해시 테이블에서 지정된 키로 키/값 페어를 변경

© 2000-2025, MetaQuotes Ltd.

Add

해시 테이블에 키/값 페어를 추가.

생성된 키/값 페어를 추가하는 버전.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // 키/값 페어
   );
```

지정된 키 및 값을 사용하여 새 키/값 페어를 추가하는 버전.

```
bool Add(
   TKey    key,                         // 키
   TValue  value                        // 값
   );
```

매개변수

pair

[in] 키/값 페어.

키

[in] 키.

값

[in] 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

정렬된 해시 테이블의 요소 수를 반환.

```
int Count();
```

© 2000-2025, MetaQuotes Ltd.

Comparer

정렬된 해시 테이블을 구성하는 데 사용되는 IComparer<T> 인터페이스에 대한 포인터를 반환.

```
IComparer<TKey>* Comparer() const;
```

값 반환

IComparer<T> 인터페이스에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

정렬된 해시 테이블 에 지정된 키/값 테이블이 포함되어 있는지 여부를 결정.

생성된 키/값 페어로 작업하기 위한 버전.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // 키/값 페어
   );
```

키/값 페어로 별도 설정된 키와 값의 형태로 작업하기 위한 버전.

```
bool Contains(
   TKey    key,                         // 키
   TValue  value                        // 값
   );
```

매개변수

item

[in] 키/값 페어.

키

[in] 키.

값

[in] 값.

값 반환

정렬된 해시 테이블에 지정된 키 /값 쌍이 포함되어 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

ContainsKey

정렬된 해시 테이블에 지정된 키가 있는 키/값 테이블이 포함되어 있는지 여부를 결정.

```
bool ContainsKey(
   TKey  key     // 키
   );
```

매개변수

키

[in] 키.

값 반환

정렬된 해시 테이블에 지정된 키를 가진 키/값 페어가 포함되어 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

ContainsValue

정렬된 해시 테이블에 지정된 값의 키/값 테이블이 포함되어 있는지 여부를 결정.

```
bool ContainsValue(
   TValue  value     // 값
   );
```

매개변수

값

[in] 값.

값 반환

정렬된 해시 테이블에 지정된 값의 키/값 페어가 포함되어 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CopyTo

정렬된 해시 테이블의 모든 키/값 페어를 지정된 인덱스부터 시작하여 시정된 배열로 복사.

키/값 페어 배열에 해시 테이블을 복사하는 버전.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // 키/값 페어 표기 배열
   const int                    dst_start=0      // 표기 시작 인덱스
   );
```

해시 테이블을 키 및 값의 개별 배열에 복사하는 버전.

```
int CopyTo(
   TKey&       dst_keys[],                       // 키 표기 배열
   TValue&     dst_values[],                     // 값 표기 배열
   const int   dst_start=0                       // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 해시 테이블의 모든 페어를 기록할 배열.

&dst_keys[]

[out] 해시 테이블의 모든 키를 기록할 배열.

&dst_values[]

[out] 해시 테이블의 모든 값이 기록될 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 키/값 페어의 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Clear

정렬된 해시 테이블에서 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

정렬된 해시 테이블에서 키/값 페어의 첫 번째 항목을 제거.

생성된 키/값 페어를 기반으로 키-값 페어를 제거하는 버전.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // 키/값 페어
   );
```

키를 기반으로 키-값 페어를 제거하는 버전.

```
bool Remove(
   TKey  key                            // 키
   );
```

매개변수

item

[in] 키/값 페어.

키

[in] 키.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TryGetValue

정렬된 해시 테이블에서 지정된 키를 가진 요소를 가져오기.

```
bool TryGetValue(
   TKey     key,      // 키
   TValue&  value     // 값 표기 변수
   );
```

매개변수

키

[in] 키.

&value

[out] 키/값 페어의 지정된 값을 쓸 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TrySetValue

지정된 키의 정렬된 해시 테이블에서 키/값 쌍을 변경.

```
bool TrySetValue(
   TKey    key,       // 키
   TValue  value      // 새 값
   );
```

매개변수

키

[in] 키.

값

[in] 지정된 키-값 페어에 할당할 새 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CSortedSet<T>

CSortedSet<T>는 ISet<T> 인터페이스를 구현하는 일반 클래스입니다.

설명

CSortedSet<T> 클래스의 유형 T의 정렬된 동적 데이터 세트를 구현이며, 각 값의 필요한 고유성을 가지고 있습니다. 이 클래스는 집합의 결합 및 교차점, 엄격하지 않은 하위 집합의 정의 등과 같은 집합 및 관련 연산을 사용하여 기본 메서드를 제공합니다.

선언

```
template<typename T>
class CSortedSet : public ISet<T>
```

헤더

```
#include <Generic\SortedSet.mqh>
```

상속 계층

ICollection
  ISet
    CSortedSet

클래스 메서드

메서드	설명
Add	정렬된 집한에 요소를 추가
Count	정렬된 집합의 요소 수를 반환
Contains	정렬된 집합에 지정된 값의 요소가 포함되어 있는지 여부를 결정
Comparer	정렬된 집합을 구성하는 데 사용되는 IComparer<T> 인터페이스에 대한 포인터를 반환
TryGetMin	정렬된 집합에서 최소 요소를 가져오기
TryGetMax	정렬된 집합에서 최대 요소를 가져오기
CopyTo	정렬된 집합의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사
Clear	정렬된 집합에서 모든 요소를 제거
Remove	정렬된 집합에서 지정한 요소의 발생을 제거
ExceptWith	현재 컬렉션과 통과된 컬랙션(배열) 간의 차이 연산을 생성
IntersectWith	현재 컬렉션과 통과된 컬렉션(배열)의 교차점 연산을 생성
SymmetricExceptWith	현재 컬렉션과 통과된 컬렉션(배열) 간의 대칭 차이 연산을 생성
UnionWith	현재 컬렉션과 통과된 컬렉션(배열)의 결합을 생성

© 2000-2025, MetaQuotes Ltd.

メソッド	説明
IsProperSubsetOf	現在 정렬된 집합이 지정된 컬렉션 또는 배열의 적절한 하위 집합인지 여부를 결정
IsProperSupersetOf	현재 정렬된 집합이 지정된 컬렉션 또는 배열의 적절한 상위 집합인지 여부를 결정
IsSubsetOf	현재 정렬된 집합이 지정된 컬렉션 또는 배열의 하위 집합인지 여부를 결정
IsSupersetOf	현재 정렬된 집합이 지정된 컬렉션 또는 배열의 상위 집합인지 여부를 결정
Overlaps	현재 정렬된 집합이 지정된 컬렉션 또는 배열과 겹치는지 여부를 결정
SetEquals	현재 정렬된 집합에 지정된 컬렉션 또는 배열의 모든 요소가 포함되어 있는지 여부를 결정
GetViewBetween	현재 정렬된 집합에서 최소값과 최대값으로 지정된 부분 집합을 가져오기
GetReverse	모든 요소가 역순으로 배열된 현재 정렬 집합의 복사본을 가져오기

© 2000-2025, MetaQuotes Ltd.

Add

정렬된 집합에 요소를 추가.

```
bool Add(
   T  value     // 요소의 값
   );
```

매개변수

값

[in] 추가할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

정렬된 집합의 요소 수를 반환.

```
int Count();
```

값 반환

요소 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

정렬된 집합에 지정된 값의 요소가 포함되어 있는지 여부를 결정.

```
bool Contains(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

지정한 값을 가진 요소가 집합에 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Comparer

정렬된 집합을 구성하는 데 사용되는 IComparer<T> 인터페이스에 대한 포인터를 반환.

```
IComparer<T>* Comparer() const;
```

값 반환

IComparer<T> 인터페이스에 대한 포인터를 반환.

© 2000-2025, MetaQuotes Ltd.

TryGetMin

정렬된 집합에서 최소 요소 가져오기

```
bool TryGetMin(
   T&  min     // 값 표기 변수
   );
```

매개변수

&min

[out] 최소값을 기록할 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

TryGetMax

정렬된 집합에서 최대 요소를 가져오기.

```
bool TryGetMax(
   T&  max     // 값 표기 변수
   );
```

매개변수

&max

[out] 최대값을 기록할 변수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CopyTo

정렬된 집합의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사.

```
int CopyTo(
   T&          dst_array[],     // 표기 배열
   const int   dst_start=0      // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 집합의 요소를 쓸 배열.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 요소 수를 반환

© 2000-2025, MetaQuotes Ltd.

Clear

정렬된 집합에서 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

정렬된 집합에서 지정된 요소의 발생을 제거.

```
bool Remove(
   T  item     // 요소 값
   );
```

매개변수

항목

[in] 삭제할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

ExceptWith

현재 컬렉션과 통과된 컬렉션(배열) 간의 차이 연산을 생성. 지정된 컬렉션(배열)에 있는 모든 요소를 현재 컬렉션(배열)에서 제거.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void ExceptWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void ExceptWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 정렬된 집합에서 제외할 집합.

&collection[]

[in] 현재 정렬된 집합에서 제외할 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

IntersectWith

현재 컬렉션과 통과된 컬렉션(배열)의 교차점 연산을 생성. 지정된 컬렉션(배열)에 있는 요소만 포함하도록 현재 컬렉션을 수정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void IntersectWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void IntersectWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 집합과 교차할 컬렉션.

&collection[]

[in] 현재 세트가 교차될 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

SymmetricExceptWith

현재 컬렉션과 통과된 컬렉션(배열) 간의 대칭 차이를 연산을 생성. 소스 개체 또는 지정된 컬렉션(배열)에 있는 요소만 포함하도록 현재 컬렉션을 수정합니다.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void SymmetricExceptWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 대칭 차이를 생성할 컬렉션.

&collection[]

[in] 대칭적 차이를 생성하는 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

UnionWith

현재 컬렉션과 통과된 컬렉션(배열)의 조합을 생성. 지정된 컬렉션(배열)에서 누락된 요소를 현재 컬렉션(배열)에 추가.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
void UnionWith(
   ICollection<T>*  collection     // 컬렉션
   );
```

배열 작업을 위한 버전.

```
void UnionWith(
   T&  array[]                     // 배열
   );
```

매개변수

collection

[in] 현재 세트가 통합될 컬렉션.

&collection[]

[in] 현재 세트가 통합될 배열.

참고

결과가 현재 컬렉션(배열)에 기록됩니다.

© 2000-2025, MetaQuotes Ltd.

IsProperSubsetOf

현재 정렬된 집합이 지정된 컬렉션 또는 배열의 적절한 하위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsProperSubsetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 정렬된 집합이 적절한 하위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IsProperSupersetOf

현재 정렬된 집합이 지정된 컬렉션 또는 배열의 적절한 상위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsProperSupersetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 정렬된 집합이 올바른 상위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IsSubsetOf

현재 정렬된 집합이 지정된 컬렉션 또는 배열의 하위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsSubsetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 정렬된 집합이 하위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

IsSupersetOf

현재 정렬된 집합이 지정된 컬렉션 또는 배열의 상위 집합인지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // 그 관계를 규명하기 위한 수집품
   );
```

배열 작업을 위한 버전.

```
bool IsSupersetOf(
   T&  array[]                     // 관계를 결정하는 배열
   );
```

매개변수

collection

[in] 관계를 결정할 컬렉션.

&collection[]

[in] 관계를 결정할 배열.

값 반환

현재 정렬된 집합이 상위 집합이면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Overlaps

현재 정렬된 집합이 지정된 컬렉션 또는 배열과 겹치는지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool Overlaps(
   ICollection<T>*  collection     // 비교할 컬렉션
   );
```

배열 작업을 위한 버전.

```
bool Overlaps(
   T&  array[]                     // 비교할 수 있는 배열
   );
```

매개변수

collection

[in] 중복 여부를 결정할 컬렉션.

&collection[]

[in] 중복 여부를 결정할 배열.

값 반환

현재 정렬된 집합과 컬렉션 또는 배열이 겹치면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

SetEquals

현재 정렬된 집합에서 지정된 컬렉션 또는 배열의 모든 요소가 포함되어 있는지 여부를 결정.

ICollection<T> 인터페이스를 구현하는 컬렉션으로 작업하기 위한 버전.

```
bool SetEquals(
   ICollection<T>*  collection     // 비교할 컬렉션
   );
```

배열 작업을 위한 버전.

```
bool SetEquals(
   T&  array[]                     // 비교할 수 있는 배열
   );
```

매개변수

collection

[in] 요소를 비교할 컬렉션.

&collection[]

[in] 요소를 비교할 배열.

값 반환

현재 정렬된 집합에 지정된 컬렉션 또는 배열의 모든 요소가 포함되어 있으면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

GetViewBetween

현재 정렬된 집합에서 최소값과 최대값으로 지정된 하위 집합 가져오기.

```
bool GetViewBetween(
   T&  array[],          // 표기 배열
   T   lower_value,      // 최소값
   T   upper_value       // 최대값
   );
```

매개변수

&array[]

[out] 부분 집합을 쓰기 위한 배열.

lower_value

[in] 범위의 최소값.

upper_value

[in] 범위의 최대값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

GetReverse

모든 요소가 역순으로 배열된 현재 정렬 세트의 복사본을 가져오기.

```
bool GetReverse(
   T&  array[]     // 표기 배열
   );
```

매개변수

&array[]

[out] 표기 배열.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

CStack<T>

CStack<T>는 ICollection<T> 인터페이스를 구현하는 일반 클래스입니다.

설명

CStack<T> 클래스는 LIFO (라스트 인, 퍼스트 아웃) 원칙에 다라 작동하는 스택으로 구성된 T 유형 데이터의 동적 컬렉션입니다.

선언

```
template<typename T>
class CStack : public ICollection<T>
```

헤더

```
#include <Generic\Stack.mqh>
```

상속 계층

ICollection
  CStack

클래스 메서드

메서드	설명
Add	스택에 요소를 추가
Count	스택의 요소 수를 반환
Contains	스택에 지정된 요소가 포함되어 있는지 여부를 결정
TrimExcess	스택의 용량을 실제 요소 수로 설정
CopyTo	스택의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사
Clear	스택에서 모든 요소를 제거
Remove	스택에서 지정한 요소의 첫 번째 발생을 제거
Push	스택에 요소를 추가
Peek	헤드 요소를 스태게서 제거하지 않고 반환
Pop	헤드 요소를 반환하고 스택에서 제거

© 2000-2025, MetaQuotes Ltd.

Add

스택에 요소를 추가.

```
bool Add(
   T  value     // 요소의 값
   );
```

매개변수

값

[in] 추가할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Count

스택의 요소 수를 반환.

```
int Count();
```

값 반환

요소 수를 반환.

© 2000-2025, MetaQuotes Ltd.

Contains

스택에 지정된 값을 가진 요소가 포함되어 있는지 여부를 결정.

```
bool Contains(
   T  item     // 검색 값
   );
```

매개변수

항목

[in] 검색된 값.

값 반환

지정한 값을 가진 요소가 스택에 있는 true를, 그렇지 않으면 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TrimExcess

스택의 용량을 실제 요소 수로 설정하여 사용하지 않는 메모리를 확보.

```
void TrimExcess();
```

© 2000-2025, MetaQuotes Ltd.

CopyTo

스택의 모든 요소를 지정된 인덱스에서 시작하여 지정된 배열로 복사.

```
int CopyTo(
   T&          dst_array[],     // 표기 배열
   const int   dst_start=0      // 표기 시작 인덱스
   );
```

매개변수

&dst_array[]

[out] 스택의 모든 요소를 지정된 인덱스에서 시장하여 지정된 배열로 복사.

dst_start=0

[in] 복사가 시작되는 배열의 인데스.

값 반환

복사된 요소 수를 반환

© 2000-2025, MetaQuotes Ltd.

Clear

스택에서 모든 요소를 제거.

```
void Clear();
```

© 2000-2025, MetaQuotes Ltd.

Remove

스택에서 지정한 요소의 첫 번째 발생을 제거.

```
bool Remove(
   T  item     // 요소 값
   );
```

매개변수

항목

[in] 삭제할 요소의 값.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Push

스택에 요소를 추가.

```
bool Push(
   T  value     // 추가할 요소element to add
   );
```

매개변수

값

[in] 추가될 요소..

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Peek

스택의 요소 수를 반환.

```
T Peek();
```

값 반환

헤드 요소를 반환.

© 2000-2025, MetaQuotes Ltd.

Pop

헤드 요소를 반환하고 스택에서 제거.

```
T Pop();
```

값 반환

헤드 요소를 반환.

© 2000-2025, MetaQuotes Ltd.

ArrayBinarySearch

요소를 비교하기 위한 IComparable<T> 인터페이스를 사용하여 오름차순으로 정렬된 1차원 배열에서 지정된 값을 검색.

```
template<typename T>
int ArrayBinarySearch(
   T&              array[],          // 검색할 배열
   const int       start_index,      // 시작 인덱스
   const int       count,            // 검색 범위
   T               value,            // 검색 값
   IComparer<T>*   comparer          // 비교할 인터페이스
   );
```

매개변수

&array[]

[out] 검색할 배열.

값

[in] 검색된 값.

**comparer*

[in] 요소 비교를 위한 인터페이스.

start_index

[in] 검색이 시작되는 시작 인덱스.

카운트

[in] 검색 범위의 길이.

값 반환

찾은 요소의 인덱스를 반환. 검색 값을 찾을 수 없으면 가장 가까운 가장 작은 요소의 인덱스를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

ArrayIndexOf

1차원 배열에서 값이 처음 나타나는 경우를 검색.

```
template<typename T>
int ArrayIndexOf(
   T&          array[],          // 검색할 배열
   T           value,            // 검색 값
   const int   start_index,      // 시작 인덱스
   const int   count             // 검색 범위
   );
```

매개변수

&array[]

[out] 검색할 배열.

값

[in] 검색된 값.

start_index

[in] 검색이 시작되는 시작 인덱스.

카운트

[in] 검색 범위의 길이.

값 반환

처음 찾은 요소의 인덱스를 반환. 값을 찾을 수 없으면 -1을 반환.

© 2000-2025, MetaQuotes Ltd.

ArrayLastIndexOf

1차원 배열에서 값이 마지막으로 발생하는지 검색.

```
template<typename T>
int ArrayLastIndexOf(
   T&          array[],          // 검색할 배열
   T           value,            // 검색 값
   const int   start_index,      // 시작 인덱스
   const int   count             // 검색 범위
   );
```

매개변수

&array[]

[out] 검색할 배열.

값

[in] 검색된 값.

start_index

[in] 검색이 시작되는 시작 인덱스.

카운트

[in] 검색 범위의 길이.

값 반환

마지막으로 찾은 요소의 인덱스를 반환. 값을 찾을 수 없으면 -1을 반환.

© 2000-2025, MetaQuotes Ltd.

ArrayReverse

1차원 배열의 요소 시퀀스를 변경.

```
template<typename T>
bool ArrayReverse(
   T&          array[],          // 소스 배열
   const int   start_index,      // 시작 인덱스
   const int   count             // 요소 수
   );
```

매개변수

&array[]

[out] 소스 배열.

start_index

[in] 시작 인덱스.

카운트

[in] 작업에 참여하는 배열 요소의 수.

값 반환

성공하면 true를, 그렇지 않으면 false를 반환.

© 2000-2025, MetaQuotes Ltd.

Compare

두 값 중 하나가 다른 값보다 "큰지, 작은지, 같은지" 여부를 비교.

두 bool 값을 비교하기 위한 버전.

```
int Compare(
   const bool  x,          // 첫 번째 값
   const bool  y           // 두 번째 값
   );
```

두 char 값을 비교하는 버전.

```
int Compare(
   const char  x,          // 첫 번째 값
   const char  y           // 두 번째 값
   );
```

두 개의 uchar 값을 비교하는 버전.

```
int Compare(
   const uchar  x,         // 첫 번째 값
   const uchar  y          // 두 버째 값
   );
```

두 개의 short 값을 비교하는 버전.

```
int Compare(
   const short  x,         // 첫 번째 값
   const short  y          // 두 번째 값
   );
```

두 가지 ushort 값을 비교하기 위한 위한 버전.

```
int Compare(
   const ushort  x,        // 첫 번째 값
   const ushort  y         // 두 번째 값
   );
```

두 색상 값을 비교하는 버전.

```
int Compare(
   const color  x,         // 첫 번째 값
   const color  y          // 두 번째 값
   );
```

두 int 값을 비교하는 버전.

```
int Compare(
   const int  x,           // 첫 번째 값
   const int  y            // 두 번째 값
   );
```

© 2000-2025, MetaQuotes Ltd.

두 uint 값을 비교하는 버전.

```
int Compare(
   const uint  x,          // 첫 번째 값
   const uint  y           // 두 번째 값
   );
```

두 datetime 값을 비교하는 버전.

```
int Compare(
   const datetime  x,      // 첫 번째 값
   const datetime  y       // 두 번째 값
   );
```

두 개의 long 값을 비교하는 버전.

```
int Compare(
   const long  x,          // 첫 번째 값
   const long  y           // 두 번째 값
   );
```

두 ulong 값을 비교하는 버전.

```
int Compare(
   const ulong  x,         // 첫 번째 값
   const ulong  y          // 두 번째 값
   );
```

두 float 값을 비교하는 버전.

```
int Compare(
   const float  x,         // 첫 번째 값
   const float  y          // 두 번째 값
   );
```

두 double 값을 비교하는 버전.

```
int Compare(
   const double  x,        // 첫 번째 값
   const double  y         // 두 번째 값
   );
```

두 문자열 값을 비교하는 버전.

```
int Compare(
   const string  x,        // 첫 번째 값
   const string  y         // 두 번째 값
   );
```

다른 유형의 두 값을 비교하는 버전.

```
template<typename T>
int Compare(
```

© 2000-2025, MetaQuotes Ltd.

```
   T  x,                  // 첫 번째 값
   T  y                   // 두 번째 값
   );
```

매개변수

x

[in] 첫 번째 값

y

[in] 두 번째 값

값 반환

비교된 두 값의 비율을 나타내는 숫자를 반환:

- 결과가 0보다 작으면, x가 y보다 작음 (x<y)
- 결과가 0이면, x가 y와 같음 (x=y)
- 결과가 0보다 크면, x가 y보다 큼 (x>y)

참고

만약 T 유형이 IComparable<T> 인터페이스를 구현하는 개체라면, 그 개체들은 Compare 메서드를 기반으로 비교될 것입니다. 다른 모든 경우에 0이 반환됩니다.

© 2000-2025, MetaQuotes Ltd.

Equals

동일성에 대한 두 값을 비교.

```
template<typename T>
bool Equals(
   T  x,      // 첫 번째 값
   T  y       // 두 번째 값
   );
```

매개변수

x

[in] 첫 번째 값

y

[in] 두 번째 값

값 반환

개체가 같으면 true를, 그렇지 않으면 false를 반환합니다.

참고

T 유형이 IEqualityComparable<T> 인터페이스를 구현하는 개체인 경우, 개체는 Equals 비교 메서드에 기초하여 비교됩니다. 동일성에 대한 표준 비교는 다른 모든 경우에 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

GetHashCode

해시 코드 값을 계산.

bool 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const bool  value          // 값
   );
```

A version for working with the char 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const char  value          // 값
   );
```

A version for working with the uchar 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const uchar  value         // 값
   );
```

short 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const short  value         // 값
   );
```

ushort 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const ushort  value        // 값
   );
```

색상 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const color  value         // 값
   );
```

int 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const int  value           // 값
   );
```

uint 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const uint  value          // 값
   );
```

datetime 유형으로 작업하기 위한 버전.

```
int GetHashCode(
```

© 2000-2025, MetaQuotes Ltd.

```
   const datetime   value     // 값
   );
```

long 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const long   value         // 값
   );
```

ulong 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const ulong   value        // 값
   );
```

float 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const float   value        // 값
   );
```

double 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const double   value       // 값
   );
```

string 유형으로 작업하기 위한 버전.

```
int GetHashCode(
   const string   value       // 값
   );
```

기타 유형으로 작업하기 위한 버전.

```
template<typename T>
int GetHashCode(
   T   value                  // 값
   );
```

매개변수

값

[in] 해시 코드를 가져올 값.

값 반환

해시 코드 반환.

참고

T 유형이 IEqualityComparable<T> 인터페이스를 구현하는 개체라면 해시 코드를 메소드를 기반으로 해시 코드 메소드를 기반으로 해시 코드를 얻습니다. 다른 모든 경우에는 해시 코드가 값 유형 이름의 해시 값으로 계산됩니다.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

파일 작업

이 섹션에는 파일 작업 클래스의 기술 세부 정보와 MQL5 표준 라이브러리의 해당 구성 요소에 대한 설명이 포함되어 있습니다.

파일 작업 클래스는 파일 입력/출력 작업을 사용하여 애플리케이션을 개발하는 시간을 절약합니다.

MQL5 표준 라이브러리(파일 작업의 경우)은 Include₩Files 폴더의 터미널 작업 디렉토리에 있습니다.

클래스	설명
CFile	기본 파일 작업 클래스
CFileBin	Binary 파일 작업 클래스
CFileTxt	텍스트 파일 작업 클래스

© 2000-2025, MetaQuotes Ltd.

CFile

CFile은 CFileBin 및 CFileTxt 클래스의 기본 클래스입니다.

설명

CFile 클래스는 모든 하위 항목에 대해 MQL5 API 파일 및 폴더에 대한 단순화된 액세스를 제공합니다.

선언

```
class CFile: public CObject
```

제목

```
#include <Files\File.mqh>
```

상속 계층

CObject

CFile

직계 후손

CFileBin, CFilePipe, CFileTxt

그룹별 클래스 메서드

특성	
Handle	파일 핸들 가져오기
Flename	파일 이름 가져오기
Flags	파일 플래그 가져오기
SetUnicode	FILE_UNICODE 플래그 설정하기/비우기
SetCommon	FILE_COMMON 플래그 설정하기/비우기
파일에 대한 일반 메서드	
Open	파일 열기
Close	파일 닫기
Delete	파일 삭제하기
IsExist	파일의 존재 여부 확인
Copy	파일 복사하기
Move	파일 이름 바꾸기/이동하기
Size	파일 크기 가져오기
Tell	현재 파일 위치 가져오기

© 2000-2025, MetaQuotes Ltd.

특성	
Seek	현재 파일 포지션 설정하기
Flush	디스크에 데이터 플러시
IsEnding	파일 끝 확인
IsLineEnding	라인 끝 확인
폴더에 대한 일반 메서드	
FolderCreate	폴더 생성
FolderDelete	폴더 삭제
FolderClean	폴더 비우기
파일 및 폴더 검색 메서드	
FileFindFirst	파일 검색 시작
FileFindNext	파일 검색 계속하기
FileFindClose	검색 핸들 닫기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Handle

열린 파일의 파일 핸들 가져오기.

```
int  Handle()
```

값 반환

클래스 인스턴스에 할당된 열린 파일의 핸들. 할당된 파일이 없으면 -1을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

FileName

열린 파일의 파일 이름 가져오기.

```
string  FileName()
```

값 반환

클래스 인스턴스에 할당된 열린 파일의 파일 이름. 할당된 파일이 없으면 ""을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Flags

열린 파일의 플래그 가져오기.

```
int  Flags()
```

값 반환

클래스 인스턴스에 할당된 열린 파일의 플래그.

© 2000-2025, MetaQuotes Ltd.

SetUnicode

FILE_UNICODE 플래그 설정하기/비우기.

```
void  SetUnicode(
   bool  unicode      // 플래그 값
  \)
```

매개변수

유니코드

[in] FILE_UNICODE 플래그의 새 값.

참고

문자열 작업의 결과는 FILE_UNICODE 플래그에 따라 달라집니다. false일 경우, ANSI 코드가 사용됩니다(1 바이트 기호). 설정된 경우 UNICODE 코드가 사용됩니다(2 바이트 기호). 파일이 이미 열려 있으면 플래그를 변경할 수 없습니다.

© 2000-2025, MetaQuotes Ltd.

SetCommon

FILE_COMMON 플래그를 설정하기/비우기.

```
void  SetCommon(
   bool  common      // 플래그 값
  \)
```

매개변수

common

[in] FILE_COMMON 플래그의 새 값.

참고

FILE_COMMON 플래그는 현재 작업 폴더를 결정합니다. 거짓이면 로컬 터미널 폴더가 현재 작업 폴더로 사용됩니다. true이면 공통 데이터 폴더가 현재 작업 폴더로 사용됩니다. 파일이 이미 열려 있으면 플래그를 변경할 수 없습니다.

© 2000-2025, MetaQuotes Ltd.

Open

지정한 파일을 열고 성공하면 클래스 인스턴스에 할당합니다.

```
int  Open(
   const string  file_name,        // 파일 이
   int           flags,            // 플래그
   short         delimiter=9       // 구분자
  \)
```

매개변수

file_name

[in] 열 파일 이름.

flags

[in] 파일 열기 플래그.

delimiter=9

[in] CSV 파일 구분자.

값 반환

열린 파일의 핸들.

참고

작업 폴더는 SetCommon() 메서드를 사용하여 이전에 설정/재설정한 플래그에 종속됩니다.

© 2000-2025, MetaQuotes Ltd.

Close

클래스 인스턴스에 할당된 파일을 닫기.

```
void  Close()
```

© 2000-2025, MetaQuotes Ltd.

Delete

파일 인스턴스에 할당된 파일을 삭제하기.

```
void  Delete()
```

Delete

지정된 파일 삭제.

```
void  Delete(
   const string  file_name      // 파일 이름
  \)
```

매개변수

file_name

[in] 삭제할 파일의 이름.

참고

작업 폴더는 SetCommon() 메서드를 사용하여 이전에 설정/재설정한 플래그에 따라 달라집니다.

© 2000-2025, MetaQuotes Ltd.

IsExist

파일의 존재 여부 확인

```
bool  IsExist(
   const string  file_name      // 파일 이름
  \)
```

매개변수

file_name

[in] 확인할 파일의 이름.

값 반환

true – 파일이 존재.

© 2000-2025, MetaQuotes Ltd.

Copy

파일 복사하기.

```
bool  Copy(
   const string  src_name,      // 파일 이름
   int           src_flag,      // 플래그
   const string  dst_name,      // 파일 이름
   int           dst_flags      // 플래그
  \)
```

매개변수

src_name
  [in] 소스 파일의 이름.

src_flag
  [in] 소스 파일의 플래그(FILE_COMMON만 사용).

dst_name
  [in] 대상 파일의 이름.

dst_flags
  [in] 대상 파일의 플래그 (FILE_REWRITE 및 FILE_COMMON만 사용).

값 반환

성공하면 true, 파일을 복사하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

Move

파일 이름 바꾸기/이동하기.

```
bool  Move(
   const string  src_name,      // 파일 이름
   int           src_flag,      // 플래그
   const string  dst_name,      // 파일 이름
   int           dst_flags      // 플래그
  \)
```

매개변수

src_name
  [in] 소스 파일 이름.

src_flag
  [in] 소스 파일 플래그(FILE_COMMON만 사용).

dst_name
  [in] 대상 파일의 이름.

dst_flags
  [in] 대상 파일의 플래그 (FILE_REWRITE 및 FILE_COMMON만 사용).

값 반환

성공하면 true, 파일 이동하기/이름 바꾸기가 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

Size

파일 크기(바이트) 가져오기.

```
ulong  Size()
```

값 반환

파일 크기(바이트). 할당된 파일이 없으면 ULONG_MAX를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Tell

현재 파일 포인터의 포지션을 가져오기.

```
ulong  Tell()
```

값 반환

현재 파일 포지션. 할당된 파일이 없는 경우, ULONG_MAX를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Seek

파일 포인터 위치 설정하기.

```
void  Seek(
   long                offset,     // 오프셋
   ENUM_FILE_POSITION  origin      // 원본
  \)
```

매개변수

오프셋

[in] 바이트 단위 파일 오프셋(음수일 수 있음).

출처

[in] 오프셋 원본.

값 반환

성공하면 true, 파일 포인터를 변경하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

Flush

디스크의 모든 파일 입력/출력 버퍼 데이터를 플러시.

```
void  Flush()
```

© 2000-2025, MetaQuotes Ltd.

IsEnding

파일 읽기 작업 중에 파일의 끝을 확인.

```
bool  IsEnding()
```

값 반환

true – 읽기 또는 탐색 작업 중에 파일의 끝에 도달했습니다.

© 2000-2025, MetaQuotes Ltd.

IsLineEnding

파일 읽기 작업 중에 텍스트 파일의 끝을 확인.

```
bool  IsLineEnding()
```

값 반환

true -txt 또는 csv 파일 읽기 작업(CR-LF chars) 중에 줄의 끝에 도달했습니다.

© 2000-2025, MetaQuotes Ltd.

FolderCreate

새 폴더 생성하기.

```
bool  FolderCreate(
   const string  folder_name      // 폴더 이름
  \)
```

매개변수

folder_name

[in] 생성할 폴더 이름. FILE_COMMON flag 플래그에 의해 정의된 폴더에 상대적인 폴더의 경로를 포함.

값 반환

성공하면 true, 폴더를 생성하지 못하면 false.

참고

작업 폴더는 SetCommon() 메서드를 사용하여 이전에 설정/재설정한 플래그에 따라 달라집니다.

© 2000-2025, MetaQuotes Ltd.

FolderDelete

지정된 폴더의 삭제.

```
bool  FolderDelete(
   const string  folder_name      // 폴더 이름
  \)
```

매개변수

folder_name

[in] 삭제할 폴더의 이름. FILE_COMMON flag 플래그에 의해 정의된 폴더에 상대적인 폴더의 경로를 포함.

값 반환

성공하면 true, 폴더를 삭제하지 못하면 false.

참고

작업 폴더는 SetCommon() 메서드를 사용하여 이전에 설정/재설정한 플래그에 따라 달라집니다.

© 2000-2025, MetaQuotes Ltd.

FolderClean

Cleans specified folder.

```
bool  FolderClean(
   const string  folder_name      // 폴더 이름
  \)
```

매개변수

folder_name

[in] 정리할 폴더의 이름. FILE_COMMON flag 플래그에 의해 정의된 폴더에 상대적인 폴더의 경로를 포함.

값 반환

성공하면 true, 폴더를 변경하지 못하면 false.

참고

작업 폴더는 SetCommon() 메서드에 의해 이전에 설정/재설정된 플래그에 따라 달라집니다.

© 2000-2025, MetaQuotes Ltd.

FileFindFirst

지정된 필터를 사용하여 파일 검색을 시작.

```
int  FileFindFirst(
   const string  filter,        // 검색 필터
   string&       file_name      // 참조
  \)
```

매개변수

필터

[in] 검색 필터.

file_name

[out] 성공할 경우 처음 찾은 파일의 이름이 문자열에 대한 참조에 배치됩니다.

값 반환

필터에 해당하는 파일이 없는 경우 FileFindNext 또는 INVALID_HANDLE을 사용하여 추가 파일 검색에 사용할 수 있는 핸들입니다.

참고

작업 폴더는 SetCommon() 메서드에 의해 이전에 설정/재설정된 플래그에 따라 달라집니다.

© 2000-2025, MetaQuotes Ltd.

FileFindNext

FileFindFirst() 메서드로 시작된 파일 검색을 계속하기.

```
bool  FileFindNext(
   int      search_handle,     // 핸들 검색
   string&  file_name          // 참조
  \)
```

매개변수

search_handle

[in] FileFindFirst() 메서드로 반환된 검색 핸들.

file_name

[in] 성공하면 찾은 파일의 이름이 배치되는 문자열에 대한 참조.

값 반환

성공하면 true, 필터에 해당하는 파일이 없는 경우 false.

© 2000-2025, MetaQuotes Ltd.

FileFindClose

검색 핸들 닫기.

```
void  FileFindClose(
   int  search_handle      // 핸들 검색
  \)
```

매개변수

search_handle

[in] FileFindFirst() 메서드로 반환된 검색 핸들.

© 2000-2025, MetaQuotes Ltd.

CFileBin

CFileBin은 binary 파일에 대한 단순 액세스를 위한 클래스입니다.

설명

CFileBin 클래스는 binary 파일에 대한 액세스를 제공합니다.

선언

```
class CFileBin: public CFile
```

제목

```
#include <Files\FileBin.mqh>
```

상속 계층

CObject
  CFile
    CFileBin

그룹별 클래스 메서드

메서드 열기	
Open	binary 파일 열기
메서드 쓰기	
WriteChar	char 또는 uchar 유형 변수 쓰기
WriteShort	short 또는 ushort 유형 변수
WriteInteger	int 또는 uint 유형 변수
WriteLong	long 또는 ulong 유형 변수
WriteFloat	float 유형 변수 쓰기
WriteDouble	double 유형 변수
WriteString	string 유형 변수 쓰기
WriteCharArray	char 또는 uchar 유형 변수의 배열 쓰기
WriteShortArray	short 또는 ushort 유형 변수 배열 쓰기
WriteIntegerArray	int 또는 uint 유형의 변수 배열 쓰기
WriteLongArray	long 또는 ulong 유형 변수 배열 쓰기
WriteFloatArray	float 변수 배열 쓰기
WriteDoubleArray	double 유형 변수 배열 쓰기
WriteObject	CObject 클래스 상속자 인스턴스의 데이터 쓰기

© 2000-2025, MetaQuotes Ltd.

메서드 열기	
메서드 읽기	
ReadChar	Reads char 또는 uchar 유형 변수 읽기
ReadShort	short 또는 ushort 유형 변수 읽기
ReadInteger	int 또는 uint 유형 변수 읽기
ReadLong	long 또는 ulong 유형 변수 읽기
ReadFloat	Reads float 유형 변수 읽기
ReadDouble	double 유형 변수 읽기
ReadString	string 유형 변수 읽기
ReadCharArray	char 또는 uchar 유형 변수 배열 읽기
ReadShortArray	short 또는 ushort 유형 변수 배열 읽기
ReadIntegerArray	int 또는 uint 유형 변수 배열 읽기
ReadLongArray	long 또는 ulong 유형 변수 배열 읽기
ReadFloatArray	float 유형 변수 배열 읽기
ReadDoubleArray	double 유형 변수 배열 읽기
ReadObject	CObject 클래스 상속자 인스턴스 데이터를 읽기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CFile에서 상속된 메서드

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

Open

지정된 binary 파일을 열고 성공하면 클래스 인스턴스에 할당합니다.

```
int  Open(
   const string  file_name,     // 파일 이름
   int           flags          // 플래그
  \)
```

매개변수

file_name

[in] 열 파일의 파일 이름.

flags

[in] 파일 열기 플래그 (FILE_BIN 플래그는 강제로 설정됨).

값 반환

열린 파일의 핸들.

© 2000-2025, MetaQuotes Ltd.

WriteChar

파일에서 char 또는 uchar 유형 변수 쓰기.

```
uint  WriteChar(
   char  value      // 값
  \)
```

매개변수

값

[in] 쓸 변수.

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteShort

파일에 short 또는 ushort 유형 변수 쓰기.

```
uint  WriteShort(
   short  value      // 값
  \)
```

매개변수

값

[in] 쓸 변수.

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteInteger

파일에 int 또는 uint 유형 변수 쓰기.

```
uint  WriteInteger(
   int  value      // 값
  \)
```

매개변수

값

[in] 쓸 변수.

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteLong

파일에 long 또는 ulong 유형 변수 쓰기.

```
uint  WriteLong(
   long  value      // 값
  \)
```

매개변수

값

[in] 쓸 변수.

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteFloat

파일에 float 유형 변수 쓰기.

```
uint  WriteFloat(
   float  value      // 값
  \)
```

매개변수

값

[in] 쓸 변수.

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteDouble

파일에 double 유형 변수 쓰기.

```
uint  WriteDouble(
   double  value      // 값
  \)
```

매개변수

값

[in] 쓸 변수.

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteString

파일에 string 유셩 변수 쓰기.

```
uint  WriteString(
   const string  value      // 값
  \)
```

매개변수

값

[in] 쓸 문자열.

값 반환

쓴 바이트 수.

WriteString

파일에 string 유셩 변수 쓰기.

```
uint  WriteString(
   const string  value,     // 값
   int           size       // 크기
  \)
```

매개변수

값

[in] 쓸 문자열.

Size

[in] 쓸 바이트 수.

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteCharArray

파일에 char 또는 uchar 유형 변수 쓰기.

```
uint  WriteCharArray(
   char&  array[],            // 배열
   int    start_item=0,       // 시작 요소
   int    items_count=-1      // 요소 수
  \)
```

매개변수

array[]

[in] 쓸 배열.

start_item=0

[in] 쓸 요소 시작.

items_count=-1

[in] 쓸 요소 수(-1 - 전체 배열).

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteShortArray

파일에 short 또는 ushort 유형 변수 배열 쓰기.

```
uint  WriteShortArray(
   short&  array[],              // 배열
   int     start_item=0,         // 시작 요소
   int     items_count=-1        // 요소 수
  \)
```

매개변수

array[]

[in] 쓸 배열.

start_item=0

[in] 쓸 요소 시작.

items_count=-1

[in] 쓸 요소 수(-1 -전체 배열).

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteIntegerArray

파일에 int 또는 uint 유형 변수 배열 쓰기.

```
uint  WriteIntegerArray(
   int&  array[],             // 배열
   int   start_item=0,        // 시작 요소
   int   items_count=-1       // 요소 수
  \)
```

매개변수

array[]

[in] 쓸 배열.

start_item=0

[in] 쓸 요소 시작.

items_count=-1

[in] 쓸 요소 수(-1 -전체 배열).

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteLongArray

파일에 long 또는 ulong 유형 변수 배열 쓰기.

```
uint  WriteLongArray(
   long&  array[],            // 배열
   int    start_item=0,       // 시작 요소
   int    items_count=-1      // 요소 수
  \)
```

매개변수

array[]

[in] 쓸 배열.

start_item=0

[in] 쓸 요소 시작.

items_count=-1

[in] 쓸 요소 수(-1 -전체 배열).

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteFloatArray

파일에 float 유형 변수 배열 쓰기.

```
uint  WriteFloatArray(
   float&  array[],              // 배열
   int     start_item=0,         // 시작 요소
   int     items_count=-1        // 요소 수
  \)
```

매개변수

array[]

[in] 쓸 배열.

start_item=0

[in] 쓸 요소 시작.

items_count=-1

[in] 쓸 요소 수(-1 -전체 배열).

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteDoubleArray

파일에 double 유형 변수 배열 쓰기.

```
uint  WriteDoubleArray(
   double&  array[],             // 쓸 배열
   int      start_item=0,        // 시작 요소
   int      items_count=-1       // 요소 수
  \)
```

매개변수

array[]

[in] 쓸 배열.

start_item=0

[in] 쓸 요소 시작.

items_count=-1

[in] 쓸 요소 수(-1 - 전체 배열).

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

WriteObject

CObject 클래스 상속자 인스턴스의 데이터를 파일에 씁니다.

```
bool  WriteObject(
   CObject*  object      // 개체에 대한 참조
  \)
```

매개변수

개체

[in] 쓸 CObject 클래스 상속자 인스턴스에 대한 참조

값 반환

성공하면 true, 데이터를 쓰지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadChar

파일에서 char 또는 uchar 유형 변수 읽기.

```
bool  ReadChar(
   char&  value      // 플래그 값
  \)
```

매개변수

값

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadShort

파일에서 short 또는 ushort 유형 변수 읽기.

```
bool  ReadShort(
   short&  value
  \)
```

매개변수

값

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadInteger

파일에서 int 또는 uint 유형 변수 읽기.

```
bool  ReadInteger(
   int&  value      // 변수
  \)
```

매개변수

값

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadLong

파일에서 long 또는 ulong 유형 변수 읽기.

```
bool  ReadLong(
   long&  value
  \)
```

매개변수

값

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadFloat

파일에서 float 유형 변수 읽기.

```
bool  ReadFloat(
   float&  value      // 변수
  \)
```

매개변수

값

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadDouble

파일에서 double 유형 변수 읽기.

```
bool  ReadDouble(
   double&  value
  \)
```

매개변수

값

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadString

파일에서 string 유형 변수 읽기.

```
bool  ReadString(
   string&  value      // 문자열
  \)
```

매개변수

값

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

ReadString

파일에서 string 유형 변수 읽기.

```
bool  ReadString(
   string&  value
  \)
```

매개변수

값

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadCharArray

파일에서 char 또는 uchar 유형 변수 배열 읽기.

```
bool  ReadCharArray(
   char&  array[],             // 배열
   int    start_item=0,        // 시작 요소
   int    items_count=-1       // 요소 수
  \)
```

매개변수

array[]

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

start_item=0

[in] 읽을 요소 시작.

items_count=-1

[in] 읽을 요소 수(파일의 끝을 읽으면 -1).

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadShortArray

파일에서 short 또는 ushort 유형 변수 배열 읽기.

```
bool  ReadShortArray(
   short&  array[],              // 배열
   int     start_item=0,         // 시작 요소
   int     items_count=-1        // 요소 수
  \)
```

매개변수

array[]

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

start_item=0

[in] 읽을 요소 시작.

items_count=-1

[in] 읽을 요소 수(파일의 끝을 읽으면 -1).

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadIntegerArray

파일에서 int 또는 uint 유형 변수 배열 읽기.

```
bool  ReadIntegerArray(
   int&  array[],             // 배열
   int   start_item=0,        // 시작 요소
   int   items_count=-1       // 요소 수
  \)
```

매개변수

array[]

[in] int 또는 uint 유형의 대상 배열에 대한 참조.

start_item=0

[in] 읽을 요소 시작.

items_count=-1

[in] 읽을 요소 수(파일의 끝을 읽으면 -1).

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadLongArray

파일에서 long 또는 ulong 유형 변수 배열 읽기.

```
bool  ReadLongArray(
   long&  array[],             // 배열
   int    start_item=0,        // 시작 요소
   int    items_count=-1       // 요소 수
  \)
```

매개변수

array[]

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

start_item=0

[in] 읽을 요소 시작.

items_count=-1

[in] 읽을 요소 수(파일의 끝을 읽으면 -1).

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadFloatArray

파일에서 float 유형 변수의 배열 읽기.

```
bool  ReadFloatArray(
   float&  array[],             // 배열
   int     start_item=0,        // 시작 요소
   int     items_count=-1       // 요소 수
  \)
```

매개변수

array[]

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

start_item=0

[in] 읽을 요소 시작.

items_count=-1

[in] 읽을 요소 수(파일의 끝을 읽으면 -1).

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadDoubleArray

파일에서 double 유형 변수 배열 읽기.

```
bool  ReadDoubleArray(
   double&  array[],             // 배열
   int      start_item=0,        // 시작 요소
   int      items_count=-1       // 요소 수
  \)
```

매개변수

array[]

[in] 읽기 데이터를 배치하기 위한 변수에 대한 참조.

start_item=0

[in] 읽을 요소 시작.

items_count=-1

[in] 읽을 요소 수(파일의 끝을 읽으면 -1).

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ReadObject

파일에서 CObject 클래스 상속자 인스턴스의 데이터 읽기.

```
bool  ReadObject(
   CObject*  object      // 포인터
  \)
```

매개변수

개체

[in] 읽을 CObject 클래스 상속자 인스턴스에 대한 포인터.

값 반환

성공하면 true, 데이터를 읽지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

CFileTxt

CFileTxtは 텍스트 파일에 대한 단순 액세스를 위한 클래스입니다.

설명

CFileTxt 클래스는 텍스트 파일에 대한 액세스를 제공합니다.

선언

```
class CFileTxt: public CFile
```

제목

```
#include <Files\FileTxt.mqh>
```

상속 계층

CObject
  CFile
    CFileTxt

그룹별 클래스 메서드

메서드 열기	
Open	텍스트 파일 열기
메서드 쓰기	
WriteString	string 유형 변수 쓰기
메서드 읽기	
ReadString	string 유형 변수 읽기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CFile에서 상속된 메서드

Handle, FileName, Flags, SetUnicode, SetCommon, Open, Close, Delete, Size, Tell, Seek, Flush, IsEnding, IsLineEnding, Delete, IsExist, Copy, Move, FolderCreate, FolderDelete, FolderClean, FileFindFirst, FileFindNext, FileFindClose

© 2000-2025, MetaQuotes Ltd.

Open

지정한 텍스트 파일을 열고 성공하면 클래스 인스턴스에 할당합니다.

```
int  Open(
   const string  file_name,     // 파일 이름
   int           flags          // 플래그
  \)
```

매개변수

file_name

[in] 열 파일 이름.

flags

[in] 파일 열기 플래그 (FILE_TXT 플래그는 강제로 설정됨).

값 반환

열린 파일 핸들.

© 2000-2025, MetaQuotes Ltd.

WriteString

파일에 string 유형 변수 쓰기.

```
uint  WriteString(
   const string  value      // 문자열
  \)
```

매개변수

값

[in] 쓸 문자열.

값 반환

쓴 바이트 수.

© 2000-2025, MetaQuotes Ltd.

ReadString

파일에서 string 유형 변수 읽기.

```
string  ReadString()
```

값 반환

읽은 문자열.

© 2000-2025, MetaQuotes Ltd.

문자열 작업

이 섹션에는 문자열 작업 클래스의 기술 세부 정보와 MQL5 스탠다드 라이브러리의 해당 구성 요소에 대한 설명이 포함되어 있습니다.

문자열 작업 클래스를 사용하는 텍스트 데이터를 처리하는 응용 프로그램을 개발하는 데 시간이 절약됩니다.

MQL5 스탠다드 라이브러리(문자열 작업의 경우)는 Include₩Strings 폴더에 있는 터미널의 작업 디렉토리에 있습니다.

클래스	설명
CString	문자열 작업 클래스

© 2000-2025, MetaQuotes Ltd.

CString

CString은 문자열 유형의 변수에 대한 단순화된 액세스를 위한 클래스입니다.

설명

CString 클래스는 문자열 변수로 작동하는 MQL5 API 함수에 대한 단순화된 액세스를 제공합니다.

선언

```
class CString: public CObject
```

제목

```
#include <Strings\String.mqh>
```

상속 계층

CObject

CString

그룹별 클래스 메서드

데이터 액세스 메서드	
Str	문자열 가져오기
Len	문자열 길이 가져오기
Copy	문자열 사본을 복사
Fill methods	
채우기	문자열 채우기
Assign	문자열을 할당
Append	문자열을 추가
Insert	문자열 삽입
메서드 배교하기	
Compare	문자열 비교
CompareNoCase	문자열을 구분하지 않는 문자열 비교를 수행
하위 문자열 메서드	
Left	왼쪽에서 하위 문자열 가져오기
Right	오른쪽에서 하위 문자열 가져오기
Mid	중간에서 하위 문자열 가져오기
다듬기/삭제하기 메서드	

© 2000-2025, MetaQuotes Ltd.

데이터 액세스 메서드	
Trim	왼쪽과 오른쪽에서 문자열 다듬기
TrimLeft	왼쪽에서 문자열 자르기
TrimRight	오른쪽에서 문자열 자르기
Clear	문자열 지우기
변환 메서드	
ToUpper	문자열을 대문자로 변환.
ToLower	문자열을 소문자로 변환.
Reverse	문자열 반전하기
검색 메서드	
Find	왼쪽에서 오른쪽으로 하위 문자열을 검색
FindRev	하위 문자열을 오른쪽에서 왼쪽으로 검색
Remove	하위 문자열 삭제
Replace	하위 문자열을 교체

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type

© 2000-2025, MetaQuotes Ltd.

Str

문자열 가져오기.

```
string  Str() const;
```

값 반환

문자열 복사.

© 2000-2025, MetaQuotes Ltd.

Len

문자열 길이 가져오기.

```
uint  Len() const;
```

값 반환

문자열 길이.

© 2000-2025, MetaQuotes Ltd.

Copy

참조별 문자열을 복사하기.

```
void  Copy(
   string&  copy      // 참조
   ) const;
```

매개변수

copy

[in] 데이터 문자열에 대한 참조.

Copy

문자열을 CString 클래스 인스턴스에 복사하기.

```
void  Copy(
   CString*  copy      // 포인터
   ) const;
```

매개변수

copy

[in] 데이터에 대한 CString 클래스 인스턴스에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

채우기

문자열을 지정된 char로 채우기.

```
bool  Fill(
   short  character      // 문자
  \)
```

매개변수

문자

[in] 문자열을 채우는 문자.

값 반환

성공하면 true, 문자열을 채우지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

Assign

문자열 할당.

```
void  Assign(
   const string  str      // 문자열
  \)
```

매개변수

str

[in] 할당할 문자열.

Assign

CString 클래스 인스턴스의 문자열을 할당.

```
void  Assign(
   CString*  str      // 포인터
  \)
```

매개변수

str

[in] 할당할 CString 클래스 인스턴스에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Append

문자열을 추가.

```
void  Append(
   const string  str      // 문자열
  \)
```

매개변수

str

[in] String to append.

Append

CString 클래스 인스턴스의 문자열을 추가.

```
void  Append(
   CString*  string      // 포인터
  \)
```

매개변수

문자열

[in] 추가할 CString 클래스 인스턴스에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Insert

지정된 위치에 문자열을 삽입하기.

```
uint  Insert(
   uint          pos,     // 포지션
   const string  str      // 문자열
  \)
```

매개변수

pos

[in] 삽입 포지션.

str

[in] 삽입할 문자열.

값 반환

결과 문자열 길이.

Insert

다음에서 지정한 위치에 문자열을 삽입: CString 클래스 인스턴스

```
uint  Insert(
   uint      pos,     // 포지션
   CString*  str      // 포인터
  \)
```

매개변수

pos

[in] 삽압할 포지션.

str

[in] 삽입할 CString 클래스 인스턴스에 대한 포인터.

값 반환

결과 문자열 길이.

© 2000-2025, MetaQuotes Ltd.

Compare

문자열 비교하기.

```
int  Compare(
   const string  str      // 문자열
   ) const;
```

매개변수

str

[in] 비교할 문자열.

값 반환

문자열이 같으면 0, 클래스 문자열이 비교할 문자열보다 낮으면 -1, 클ㄹ스 문자열이 비교할 문자열보다 크면 1.

Compare

다음과 비교: CString 클래스 인스턴트 문자열.

```
int  Compare(
   CString*  str      // 포인터
   ) const;
```

매개변수

str

[in] 비교할 CString 클래스 인스턴스에 대한 포인터.

값 반환

문자열이 같으면 0, 클래스 문자열이 비교할 문자열보다 낮으면 -1, 클래스 문자열이 비교할 문자열보다 크면 1.

© 2000-2025, MetaQuotes Ltd.

CompareNoCase

대소문자를 구분하지 않는 문자열 비교를 수행.

```
int  CompareNoCase(
   const string  str      // 문자열
   ) const;
```

매개변수

str

[in] 비교할 문자열.

값 반환

문자열이 같으면 0, 클래스 문자열이 비교할 문자열보다 낮으면 -1, 클래스 문자열이 비교할 문자열보다 크면 1.

CompareNoCase

다음과 비교: 대소문자를 구분하지 않는 문자열을 CString 클래스 인스턴스 문자열.

```
int  CompareNoCase(
   CString*  str      // 포인터
   ) const;
```

매개변수

str

[in] 비교할 CString 클래스 인스턴스에 대한 포인터.

값 반환

문자열이 같으면 0, 클래스 문자열이 비교할 문자열보다 낮으면 -1, 크래스 문자열이 비교할 문자열보다 크면 1.

© 2000-2025, MetaQuotes Ltd.

Left

문자열의 시작 부분에서 지정한 길이의 하위 문자열을 가져오기.

```
string  Left(
   uint  count      // 길이
  \)
```

매개변수

카운트

[in] 하위 문자열 길이.

값 반환

결과 하위 문자열.

© 2000-2025, MetaQuotes Ltd.

Right

문자열 끝에서 지정한 길이의 하위 문자열 가져오기.

```
string  Right(
   uint  count      // 길이
  \)
```

매개변수

카운트

[in] 하위 문자열 길이.

값 반환

결과 하위 문자열.

© 2000-2025, MetaQuotes Ltd.

Mid

지정된 문자열 위치에서 지정한 길이의 하위 문자열을 가져오기.

```
string  Mid(
   uint  pos,        // 포지션
   uint  count       // 길이
  \)
```

매개변수

pos

[in] 하위 문자열 포지션.

카운트

[in] 하위 문자열 길이.

값 반환

결과 하위 문자열.

© 2000-2025, MetaQuotes Ltd.

Trim

문자열의 양쪽 끝에 있는 집합 내의 모든 문자(' ','\t','\r','\n')를 이 문자열에서 제거하기.

```
int  Trim(
   const string  targets      // 집합
  \)
```

매개변수

목표

[in] 제거할 문자 집합.

값 반환

제거된 문자 수.

예시:

```
//--- CString::Trim의 예
#include <Strings\String.mqh>
//---
void OnStart()
  {
   CString str;
   //---
   str.Assign("   \t\tABCD\r\n");
   printf("소스 문자열 '%s'",str.Str());
   //---
   str.Trim("DA-DA-DA");
   printf("결과 문자열 '%s'",str.Str());
  }
```

© 2000-2025, MetaQuotes Ltd.

TrimLeft

문자열의 시작 부분에 있는 집합 내의 모든 문자(' ','\t','\r','\n')를 이 문자열에서 제거하기.

```
int  TrimLeft(
   const string  targets      // 집합
  \)
```

매개변수

목표

[in] 제거할 문자 집합.

값 반환

제거된 문자 수.

© 2000-2025, MetaQuotes Ltd.

TrimRight

문자열의 끝 부분에 있는 집합 내의 모든 문자(' ','\t','\r','\n')를 이 문자열에서 제거하기.

```
int  TrimRight(
   const string  targets      // 집합
  \)
```

매개변수

목표

[in] 제거할 문자 집합.

값 반환

제거된 문자 수.

© 2000-2025, MetaQuotes Ltd.

Clear

문자열 지우기.

```
bool  Clear()
```

값 반환

성공하면 true, 문자열을 지우지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ToUpper

모든 문자열 문자를 대문자로 변환.

```
bool  ToUpper()
```

값 반환

성공하면 true, 대문자로 변환하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ToLower

모든 문자열 문자를 소문자로 변환.

```
bool  ToLower()
```

값 반환

성공하면 true, 소문자로 변환하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

Reverse

문자열 반전하기 (초기 문자와 마지막 문자는 페어로 위치를 교환).

```
void  Reverse()
```

© 2000-2025, MetaQuotes Ltd.

Find

지정한 위치에서 하위 문자열의 첫 번째 일치 항목을 검색하기.

```
int  Find(
   uint          start,          // 포지션
   const string  substring       // 하위 문자열
   ) const;
```

매개변수

시작

[in] 하위 문자열 검색을 위한 초기 위치.

하위 문자열

[in] 검색할 샘플 하위 문자열.

값 반환

하위 문자열의 첫 번째 일치 인덱스 (하위 문자열을 찾을 수 없으면 -1).

© 2000-2025, MetaQuotes Ltd.

FindRev

하위 문자열의 마지막 일치 항목을 검색.

```
int  FindRev(
   const string  substring      // 하위 문자열
   ) const;
```

매개변수

하위 문자열

[in] 검색할 샘플 하위 문자열.

값 반환

하위 문자열의 마지막 일치 인덱스 (하위 문자열이 발견되지 않으면 -1).

© 2000-2025, MetaQuotes Ltd.

Remove

모든 하위 문자열 일치를 제거.

```
uint  Remove(
   const string  substring      // 하위 문자열
  \)
```

매개변수

하위 문자열

[in] 검색할 샘플 하위 문자열.

값 반환

하위 문자열 제거 수.

© 2000-2025, MetaQuotes Ltd.

Replace

다음을 교체: 모든 하위 문자열 일치.

```
uint  Replace(
   const string  substring,     // 하위 문자열
   const string  newstring      // 하위 문자열
  \)
```

매개변수

하위 문자열

[in] 검색할 샘플 하위 문자열.

newstring

[in] 교체할 샘플 하위 문자열.

값 반환

하위 문자열 교체 횟수.

© 2000-2025, MetaQuotes Ltd.

그래픽 개체

이 섹션에는 그래픽 개체 클래스 작업에 대한 기술 세부 정보와 MQL5 표준 라이브러리 관련 구성 요소에 대한 설명이 포함되어 있습니다.

그래픽 개체 클래스를 사용하면 사용자 지정 프로그램(스크립트, 엑스퍼트 어드바이저)을 만들 때 시간이 절약됩니다.

MQL5 표준 라이브러리(그래픽 개체의 경우)는 Include₩ChartObjects 폴더의 터미널의 디렉토리에 있습니다.

클래스/그룹	설명
CChartObject	그래픽 개체의 기본 클래스
Line	그룹 클래스 "선"
채널	그룹 클래스 "채널"
갠(Gann) 도구	그룹 클래스 "갠(Gann)"
피보나치 도구	그룹 클래스 "피보나치"
엘리엇 도구	그룹 클래스 "엘리엇"
모양	그룹 클래스 "모양"
화살표	그룹 클래스 "화살표"
컨트롤	그룹 플래스 "컨트롤"

© 2000-2025, MetaQuotes Ltd.

CChartObject

CChartObject는 표준 MQL5 라이브러리의 차트 그래픽 개체 클래스에 대한 기본 클래스입니다.

설명

CChartObject 클래스는 모든 하위 항목에 대해 MQL5 API 함수에 대한 단순화된 액세스를 제공합니다.

선언

```
class CChartObject : public CObject
```

제목

```
#include <ChartObjects\ChartObject.mqh>
```

상속 계층

CObject

CChartObject

직계 후손

CChartObjectArrow, CChartObjectBitmap, CChartObjectBmpLabel, CChartObjectCycles, CChartObjectElliottWave3, CChartObjectEllipse, CChartObjectFiboArc, CChartObjectFiboFan, CChartObjectFiboTimes, CChartObjectHLine, CChartObjectRectangle, CChartObjectSubChart, CChartObjectText, CChartObjectTrend, CChartObjectTriangle, CChartObjectVLine

그룹별 클래스 메서드

특성	
ChartId	그래픽 개체가 속한 차트의 ID 가져오기
Window	그래픽 개체가 있는 차트 창의 수 가져오기
Name	그래픽 개체 이름을 가져오기/설정하기
NumPoints	고정점 수 가져오기
Assign	
Attach	차트 그래픽 개체 바인딩
SetPoint	고정점 매개변수 설정
Delete	
Delete	차트 그래픽 개체를 삭제
Detach	차트 그래픽 개체를 분리
Shift	
ShiftObject	상대 개체 이동
ShiftPoint	상대 개체 점 이동

© 2000-2025, MetaQuotes Ltd.

특성	
개체 속성	
시간	개체 점의 시간 좌표를 가져오거나 설정
가격	개체 점의 가격 좌표를 가져오기/설정하기
Color	개체 색을 가져오기/설정하기
스타일	개체 선 스타일 가져오기/설정하기
Width	개체 선 너비 가져오기/설정하기
BackGround	배경에 개체 그리기 플래그 가져오기/설정하기
Selected	"선택한" 플래그를 가져오기/설정하기
Selectable	선택 가능한 개체 플래그를 가져오기/설정하기
설명	개체의 텍스트 가져오기/설정하기
Tooltip	개체의 도구 설명을 가져오기/설정하기
Timeframes	개체 표시 플래그 마스크 가져오기/설정하기
Z_Order	차트에서 마우스를 클릭하는 이벤트를 수신하기 위한 그래픽 개체 우선 순위를 가져오기/설정하기
CreateTime	개체 생성 시간을 가져오기
개체 수준 속성	
LevelsCount	개체 수준 수를 가져오기/설정하기
LevelColor	수준 선 색상 가져오기/설정하기
LevelStyle	수준 선 스타일 가져오기/설정하기
LevelWidth	수준 선 너비 가져오기/설정하기Gets/sets the level line width
LevelValue	수준 가져오기/설정하기
LevelDescription	수준 텍스트 가져오기/설정하기
API MQL5 함수에 엑세스	
GetInteger	개체 속성 값을 가져오기
SetInteger	개체 속성 값 설정하기
GetDouble	개체 속성 값을 가져오기
SetDouble	개체 속성 값 설정하기
GetString	개체 속성 값을 가져오기
SetString	개체 속성 값 설정하기
입력/출력	
virtual Save	파일에 쓰기 가상 메서드

© 2000-2025, MetaQuotes Ltd.

特성	
virtual Load	파일에서 읽기 가상 메서드
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

ChartId

그래픽 개체가 속한 차트의 ID를 가져오기.

```
long  ChartId() const
```

값 반환

그래픽 개체가 있는 차트의 ID. 바인딩된 개체가 없으면 -1을 반환합니다.

예시:

```
//--- CChartObject::ChartId 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 차트 식별자 가져오기
   long chart_id=object.ChartId();
  }
```

© 2000-2025, MetaQuotes Ltd.

Window

그래픽 개체가 있는 차트 창 인덱스 가져오기.

```
int  Window() const
```

값 반환

그래픽 개체가 있는 차트 창 숫자 (0 - 주 창). 바인딩된 개체가 없으면 -1을 반환합니다.

예시:

```
//--- CChartObject::Window 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 창 가져오기
   int window=object.Window();
  }
```

© 2000-2025, MetaQuotes Ltd.

Name (Get Method)

그래픽 개체 이름 가져오기.

```
string  Name() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 이름. 연결된 개체가 없으면 NULL을 반환합니다.

Name (Set Method)

그래픽 개체 이름 설정하기.

```
bool  Name(
   string  name      // 새 이름
  \)
```

매개변수

name

[in] 그래픽 개체의 새 이름.

값 반환

성공하면 true, 이름을 변경하지 못하면 false.

예시:

```
//--- example for CChartObject::Name
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 이름 가져오기
   string object_name=object.Name();
   if(object_name!="MyChartObject")
     {
     //--- 차트 개체 이름 설정하기
     object.Name("MyChartObject");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

NumPoints

그래픽 개체의 고정점 수 가져오기.

```
int  NumPoints() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 연결점 수. 연결된 개체가 없으면, 0을 반환합니다.

예시:

```
//--- CChartObject::NumPoints 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 점 수 가져오기
   int points=object.NumPoints();
  }
```

© 2000-2025, MetaQuotes Ltd.

Attach

그래픽 개체를 클리스의 인스턴스에 연결.

```
bool  Attach(
   long     chart_id,      // 차트 ID
   string   name,          // 개체의 이름
   int      window,        // 차트 창
   int      points         // 포인트 수
  \)
```

매개변수

chart_id

[out] 차트 식별자.

name

[in] 그래픽 개체의 이름.

window

[in] 차트 창 숫자 (0 – 주 창).

points

[in] 그래픽 개체의 고정점 수.

값 반환

성공하면 true, 개체를 바인딩할 수 없으면 false.

예시:

```
//--- CChartObject::Attach 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 연결
   if(!object.Attach(ChartID(),"MyObject",0,2))
     {
      printf("개체 연결 오류");
      return;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SetPoint

그래픽 개체의 지정된 고정점의 새 좌표를 설정하기.

```
bool  SetPoint(
   int       point,          // 점 수
   datetime  new_time,       // 시간 좌표
   double    new_price       // 가격 좌표
  \)
```

매개변수

점

[in] 그래픽 개체 고정점의 수.

new_time

[in] 지정된 고정점의 시간 좌표에 대한 새 값.

new_price

[in] 지정된 고정점의 가격 좌표에 대한 새 값.

값 반환

성공하면 true, 점 좌표를 변경할 수 없으면 false.

예시:

```
//--- CChartObject::SetPoint 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   if(object.NumPoints()>0)
     {
      //--- 차트 개체 점 설정
      object.SetPoint(0,CurrTime(),price);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Delete

차트에서 연결된 그래픽 개체를 제거.

```
bool  Delete()
```

값 반환

성공하면 true, 개체를 제거할 수 없으면 false.

예시:

```
//--- CChartObject::Delete 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 분리
   if(!object.Delete())
     {
      printf("개체 삭제 오류");
      return;
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Detach

그래픽 개체 분리.

```
void  Detach()
```

값 반환

없음.

예시:

```
//--- CChartObject::Detach 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 분리
   object.Detach();
  }
```

© 2000-2025, MetaQuotes Ltd.

ShiftObject

그래픽 개체 이동.

```
bool  ShiftObject(
   datetime  d_time,      // 시간 좌표의 증가
   double    d_price      // 가격 좌표의 증가
  \)
```

매개변수

d_time

[in] 모든 고정점의 시간 좌표 증가.

d_price

[in] 모든 고정점의 가격 좌표 증가.

값 반환

성공하면 true, 개체를 이동하지 못하면 false.

예시:

```
//--- CChartObject::ShiftObject 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //--- 시프트 차트 개체
   object.ShiftObject(d_time,d_price);
  }
```

© 2000-2025, MetaQuotes Ltd.

ShiftPoint

그래픽 개체의 지정된 고정점을 이동하기.

```
bool  ShiftPoint(
   int       point,       // 점 수
   datetime  d_time,      // 시간 좌표의 증가
   double    d_price      // 가격 좌표의 증가
  \)
```

매개변수

점

[in] 그래픽 개체 고정점 수.

d_time

[in] 지정된 점의 시간 좌표 증가.

d_price

[in] 지정된 점의 가격 좌표 증가.

값 반환

성공하면 true, 점을 이동하지 못하면 false.

예시:

```
//--- CChartObject::ShiftPoint 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //---
   if(object.NumPoints()>0)
     {
      //--- 차트 개체의 이동점
      object.ShiftPoint(0,d_time,d_price);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Time (Get Method)

그래픽 개체의 지정된 고정점 시간 좌표 가져오기.

```
datetime  Time(
   int  point      // 점 수
   ) const
```

매개변수

점

[in] 그래픽 개체 고정점 수.

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 지정된 고정점 시간 좌표. 연결된 개체가 없거나 개체에 이 점이 없으면 0을 반환합니다.

Time (Set Method)

그래픽 개체의 지정된 앵커 점의 시간 좌표를 설정하기.

```
bool  Time(
   int       point,        // 점 수
   datetime  new_time      // 시간
  \)
```

매개변수

점

[in] 그래픽 개체 고정점 수.

new_time

[in] 지정된 그래픽 개체 고정점의 시간 좌표에 대한 새 값.

값 반환

성공하면 true, 시간 좌표를 변경하지 못하면 false.

예시:

```
//--- CChartObject::Time 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- 차트 개체 포인트 시간 가져오기
      datetime point_time=object.Time(i);
      if(point_time==0)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- 차트 개체 포인트 시간 설정하기
          object.Time(i,TimeCurrent());
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

Price (Get Method)

그래픽 개체의 지정된 고정점 가격 좌표 가져오기.

```
double  Price(
   int  point      // 점 수
   ) const
```

매개변수

점

[in] 그래픽 개체 고정점 수.

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 지정된 고정점 가격 좌표. 연결된 개체가 없거나 개체에 이 점이 없으면 EMPTY_VALUE을 반환합니다.

Price (Set Method)

그래픽 개체의 지정된 고정점 가격 좌표를 설정하기.

```
bool  Price(
   int     point,          // 점 수
   double  new_price       // 가격
  \)
```

매개변수

점

[in] 그래픽 개체 고정점 수.

new_price

[in] 지정된 그래픽 개체 고정점의 가격 좌표에 대한 새 값.

값 반환

성공하면 true, 가격 좌표를 변경하지 못하면 false.

예시:

```
//--- CChartObject::Price 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- 차트 개체 점 가격 가져오기
      double point_price=object.Price(i);
```

© 2000-2025, MetaQuotes Ltd.

```
      if(point_price!=price)
        {
         //--- 차트 개체 점 가격 설정하기
         object.Price(i,price);
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Color (Get Method)

그래픽 개체의 선 색 가져오기.

```
color  Color() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 선 색상. 개체가 연결되어 있지 않으면 CLR_NONE을 반환합니다.

Color (Set Method)

그래픽 개체의 선 색상을 설정.

```
bool  Color(
   color  new_color      // 새 색상
  \)
```

매개변수

new_color

[in] 그래픽 개체 선 색상의 새 값.

값 반환

성공하면 true, 색상을 변경하지 않으면 false.

예시:

```
//--- CChartObject::Color 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 색상 가져오기
   color object_color=object.Color();
   if(object_color!=clrRed)
     {
     //--- 차트 개체 색상 설정하기
     object.Color(clrRed);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Style (Get Method)

그래픽 개체의 선 스타일 가져오기.

```
ENUM_LINE_STYLE  Style() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 선 스타일. 연결된 개체가 없으면 WRONG_VALUE를 반환합니다.

Style (Set Method)

그래픽 개체의 선 스타일 설정하기.

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // 스타일
  \)
```

매개변수

new_style

[in] 그래픽 개체 선 스타일의 새 값.

값 반환

성공하면 true, 스타일을 변경하지 못하면 false.

예시:

```
//--- CChartObject::Style 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 스타일 가져오기
   ENUM_LINE_STYLE style=object.Style();
   if(style!=STYLE_SOLID)
     {
      //--- 차트 개체 스타일 설정하기
      object.Style(STYLE_SOLID);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Width (Get Method)

그래픽 개체 선 버비 가져오기.

```
int  Width() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 선 너비. 연결된 개체가 없으면 -1을 반환합니다.

Width (Set Method)

그래픽 개체의 선 너비를 설정하기.

```
bool  Width(
   int  new_width      // 두께
  \)
```

매개변수

new_width

[in] 그래픽 개체 선 너비의 새 값.

값 반환

성공하면 true, 너비를 변경하지 못하면 false.

예시:

```
//--- CChartObject::Width 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 너비 가져오기
   int width=object.Width();
   if(width!=1)
     {
      //--- 차트 개체 너비 설정하기
      object.Width(1);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

S배경 (메서드 가져오기)

배경에 그래픽 개체를 그리는 플래그를 가져오기.

```
bool  Background() const
```

값 반환

배경에 클래스 인스턴스에 연결된 그래픽 개체를 그리기 위한 플래그. 연결된 개체가 없으면 false를 반환합니다.

Background (Set Method)

배경에 그래픽 개체를 그리는 플래그를 설정.

```
bool  Background(
   bool  background      // 플래그의 값
  \)
```

매개변수

배경

[in] 배경에 그래픽 개체를 그리기 위한 플래그의 새 값.

값 반환

성공하면 true, 플래그를 변경할 수 없으면 false.

예시:

```
//--- CChartObject::Background 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 배경 플래그 가져오기
   bool background_flag=object.Background();
   if(!background_flag)
     {
     //--- 차트 개체의 배경 플래그 설정하기
     object.Background(true);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Selected (Get Method)

그래픽 개체가 선택되었음을 나타내는 플래그 가져오기. 즉, 그래픽 개체가 선택되어 있는지 여부.

```
bool  Selected() const
```

값 반환

클래스 인스턴스에 연결된 개체가 선택된 상태입니다. 연결된 개체가 없으면 false를 반환합니다.

Selected (Set Method)

그래픽 개체가 선택되었음을 나타내는 플래그를 설정하기.

```
bool  Selected(
   bool  selected      // 플래그 값
  \)
```

매개변수

selected

[in] 그래픽 개체가 선택되었음을 나타내는 플래그의 새 값.

값 반환

성공하면 true, 플래그를 변경할 수 없으면 false.

예시:

```
//--- CChartObject::Selected 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 "선택된" 플래그 가져오기
   bool selected_flag=object.Selected();
   if(selected_flag)
     {
     //--- 차트 개체의 "선택된" 플래그 설정하기
     object.Selected(false);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Selectable (Get Method)

선택할 그래픽 개체의 기능을 나타내는 플래그 가져오기. 즉, 그래픽 개체를 선택할 수 있는지 여부.

```
bool  Selectable() const
```

값 반환

선택할 클래스의 인스턴스에 연결된 그래픽 개체의 기능을 나타내는 플래그. 연결된 개체가 없으면 false를 반환.

Selectable (Set Method)

선택할 그래픽 개체의 기능을 나타내는 플래그를 설정하기.

```
bool  Selectable(
   bool  selectable      // 플래그의 값
  \)
```

매개변수

selectable

[in] 선택할 그래픽 개체의 기능을 나타내는 플래그의 새 값.

값 반환

성공하면 true, 플래그를 변경할 수 없으면 false.

예시:

```
//--- CChartObject::Selectable 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 "선택할" 플래그 가져오기
   bool selectable_flag=object.Selectable();
   if(selectable_flag)
     {
     //--- 차트 개체의 "선택할" 플래그 설정하기
     object.Selectable(false);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Description (Get Method)

그래픽 개체의 설명(텍스트) 가져오기.

```
string  Description() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체에 대한 설명(텍스트). 연결된 개체가 없으면 NULL을 반환합니다.

Description (Set Method)

그래픽 개체에 대한 설명(텍스트)을 설정.

```
bool  Description(
   string  text      // 텍스트
  \)
```

매개변수

텍스트

[in] 그래픽 개체에 대한 새 설명(텍스트).

값 반환

성공하면 true, 설명(텍스트)을 변경하지 못하면 false.

예시:

```
//--- CChartObject::Description 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 설명 가져오기
   string description=object.Description();
   if(description=="")
     {
      //--- 차트 개체 설명 설정하기
      object.Description("MyObject");
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Tooltip (Get Method)

그래픽 개체 도구 설명 텍스트 가져오기.

```
string  Tooltip() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 도구 설명 텍스트. 연결된 개체가 없으면 NULL을 반환합니다.

Tooltip (Set Method)

그래픽 개체의 도구 설명 텍스트 설정하기.

```
bool  Tooltip(
   string  new_tooltip      // 도구 설명의 새 텍스트
  \)
```

매개변수

new_tooltip

[in] 그래픽 개체 도구 설명의 새 텍스트.

값 반환

성공하면 true, 도구 설명을 변경하지 못하면 false.

참고:

속성을 설정하지 않은 경우 터미널에서 자동으로 설정된 도명구 설이 표시됩니다. 도구 설명은 "₩" (라인 피트) 값을 설정하여 비활성화할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

Timeframes (Get Method)

그래픽 개체의 가시성 플래그 가져오기.

```
int  Timeframes() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 가시성 플래그. 연결된 개체가 없으면, 0을 반환합니다.

Timeframes (Set Method)

그래픽 개체의 플래그 가시성 설정.

```
bool  Timeframes(
   int  new_timeframes      // 가시성 플래그
  \)
```

매개변수

new_timeframes

[in] 그래픽 개체의 새 가시성 플래그.

값 반환

성공하면 true, 가시성 플래그를 변경할 수 없으면 false.

예시:

```
//--- CChartObject::Timeframes 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 시간 프레임 가져오기
   int timeframes=object.Timeframes();
   if(!(timeframes&OBJ_PERIOD_H1))
     {
      //--- 차트 개체 시간 프레임 설정하기
      object.Timeframes(timeframes|OBJ_PERIOD_H1);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Z_Order (Get Method)

차트에서 마우스를 클릭하는 이벤트를 수신하기 위한 그래픽 개체 우선 순위를 가져오기 (CHARTEVENT_CLICK).

```
long  Z_Order() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 우선 순위. 개체가 연결되어 있지 않으면 0을 반환합니다.

Z_Order (Set Method)

차트에서 마우스를 클릭하는 이벤트를 수신하기 위한 그래픽 개체 우선 순위를 설정 (CHARTEVENT_CLICK).

```
bool  Z_Order(
   long  value      // 그래픽 개체 우선 순위
  \)
```

매개변수

값

[in] New value of priority of a graphical object for receiving the event CHARTEVENT_CLICK 이벤트를 수신하기 위한 그래픽 개체의 새 우선 순위 값.

값 반환

성공하면 true, 우선 순위를 변경하기 못하면 false.

참고

Z_Order 속성은 그래픽 개체에 대한 클릭을 처리할 때 우선 순위를 관리합니다. 0(기본 값)보다 큰 값을 설정하여 마우스 클릭을 처리할 때 개체 우선 순위를 높일 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

CreateTime

그래픽 개체 생성 시간을 가져오기.

```
datetime  CreateTime() const
```

값 반환

클래스의 인스턴스에 연결된 그래픽 개체의 생성 시간. 연결된 개체가 없으면, 0을 반환합니다.

예시:

```
//--- CChartObject::CreateTime 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체 생성 시간 가져오기
   datetime create_time=object.CreateTime();
  }
```

© 2000-2025, MetaQuotes Ltd.

LevelsCount (Get Method)

그래픽 개체의 레벨 수 가져오기.

```
int  LevelsCount() const
```

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 레벨 수. 연결된 개체가 없으면, 0을 반환합니다.

LevelsCount (Set Method)

그래픽 개체의 레벨 수를 설정.

```
bool  LevelsCount(
   int  levels      // 레벨 숫자
  \)
```

매개변수

levels

[in] 그래픽 개체의 새 레벨 수.

값 반환

성공하면 true, 레벨 수를 변경할 수 없으면 false.

예시:

```
//--- CChartObject::LevelsCount 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 레벨 수 가져오기
   int levels_count=object.LevelsCount();
   //--- 차트 개체의 레벨 수 설정하기
   object.LevelsCount(levels_count+1);
  }
```

© 2000-2025, MetaQuotes Ltd.

LevelColor (Get Method)

그래픽 개체의 지정된 수준의 선 색상을 가져오기.

```
color  LevelColor(
   int  level      // 레벨 숫자
   ) const
```

매개변수

수준

[in] 그래픽 개체 수준 숫자.

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 지정된 수준 선 색상. 연결된 개체가 없거나 개체에 지정된 수준이 없으면 CLR_NONE을 반환합니다.

LevelColor (Set Method)

그래픽 개체의 지정된 수준의 선 색을 설정하기.

```
bool  LevelColor(
   int    level,          // 레벨 숫자
   color  new_color       // 새 색상
  \)
```

매개변수

수준

[in] 그래픽 개체 수준 숫자.

new_color

[in] 그래픽 개체의 지정된 수준의 새 선 색상.

값 반환

성공하면 true, 색상을 변경하지 않으면 false.

예시:

```
//--- CChartObject::LevelColor 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 차트 개체의 수준 색상 가져오기
      color level_color=object.LevelColor(i);
      if(level_color!=clrRed)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- 차트 개체의 수준 색상 설정하기
          object.LevelColor(i,clrRed);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelStyle (Get Method)

그래픽 개체의 지정된 수준의 선 스타일 가져오기.

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // 레벨 숫자
   ) const
```

매개변수

수준

[in] 그래픽 개체 수준 숫자.

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 지정된 수준의 선 스타일. 연결된 개체가 없거나 개체에 지정된 수준이 없는 경우 WRONG_VALUE를 반환합니다.

LevelStyle (Set Method)

그래픽 개체의 지정된 수준의 선 스타일을 설정.

```
int  LevelStyle(
   int              level,     // 레벨 숫자
   ENUM_LINE_STYLE  style      // 선 스타일
  \)
```

매개변수

수준

[in] 그래픽 개체 수준 숫자.

스타일

[in] 그래픽 개체의 지정된 수준의 새 선 스타일입니다.

값 반환

성공하면 true, 스타일을 변경하지 못하면 false.

예시:

```
//--- CChartObject::LevelStyle 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 차트 개체의 수준 스타일 가져오기
      ENUM_LINE_STYLE level_style=object.LevelStyle(i);
      if(level_style!=STYLE_SOLID)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- 차트 개체의 수준 스타일 설정하기
          object.LevelStyle(i,STYLE_SOLID);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelWidth (Get Method)

그래픽 개체의 지정된 수준의 선 너비를 가져오기.

```
int  LevelWidth(
   int  level      // 레벨 숫자
   ) const
```

매개변수

수준

[in] 그래픽 개체 수준 숫자.

값 반환

클래스 인스턴스에 연결된 그래픽 개체의 지정된 수준 선 너비. 연결된 개체가 없거나 개체에 지정된 수준이 없으면 -1을 반환합니다.

LevelWidth (Set Method)

그래픽 개체의 지정된 수준 선 너비를 설정.

```
bool  LevelWidth(
   int  level,          // 레벨 숫자
   int  new_width       // 새 너비
  \)
```

매개변수

수준

[in] 그래픽 개체 수준 숫자.

new_width

[in] 그래픽 개체의 지정된 수준의 새 선 너비.

값 반환

성공하면 true, 너비를 변경하지 못하면 false.

예시:

```
//--- example for CChartObject::LevelWidth
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 차트 개체 수준 너비 가져오기
      int level_width=object.LevelWidth(i);
      if(level_width!=1)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- 차트 개체 수준 너비 설정하기
          object.LevelWidth(i,1);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelValue (Get Method)

그래픽 개체의 지정된 수준 값을 가져오기.

```
double  LevelValue(
   int  level      // 레벨 숫자
   ) const
```

매개변수

수준

[in] 그래픽 개체 레벨 숫자.

값 반환

클래스 인스턴스에 바인딩된 그래픽 개체의 지정된 수준 값. 바인딩된 개체가 없거나 개체가 수준을 지정하지 않은 경우 EMPTY_VALUE를 반환합니다.

LevelValue (Set Method)

그래픽 개체의 지정된 수준 값을 설정.

```
bool  LevelValue(
   int     level,          // 레벨 숫자
   double  new_value       // 새 값
  \)
```

매개변수

수준

[in] 그래픽 개체 레벨 숫자.

new_value

[in] 그래픽 개체의 지정된 수준의 새 값.

값 반환

성공하면 true, 값을 변경하지 못하면 false.

예시:

```
//--- CChartObject::LevelValue 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 차트 개체의 수준 값 가져오기
      double level_value=object.LevelValue(i);
      if(level_value!=0.1*i)
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- 차트 개체 수준 값 설정
          object.LevelValue(i,0.1*i);
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

LevelDescription (Get Method)

그래픽 개체 수준에 대한 설명(텍스트)을 가져오기.

```
string  LevelDescription(
   int  level       // 레벨 숫자
   ) const
```

매개변수

수준

[in] 그래픽 개체 레벨 숫자.

값 반환

클래스 인스턴스에 바인딩된 그래픽 개체의 지정된 수준에 대한 설명(텍스트). 바인딩된 개체가 없거나 개체에 지정된 수준이 없는 경우 NULL을 반환.

LevelDescription (Set Method)

지정한 그래픽 개체 수준에 대한 설명(텍스트)을 설정.

```
bool  LevelDescription(
   int     level ,     // 레벨 숫자
   string  text        // 텍스트
  \)
```

매개변수

수준

[in] 그래픽 개체 레벨 숫자.

텍스트

[in] 지정된 그래픽 개체 수준의 새 설명 값(텍스트).

값 반환

성공하면 true, 설명(텍스트)를 변경하지 못하면 false.

예시:

```
//--- CChartObject::LevelDescription 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 차트 개체 수준 설명 가져오기
      string level_description=object.LevelDescription(i);
      if(level_description=="")
```

© 2000-2025, MetaQuotes Ltd.

```
         {
          //--- 차트 개체 수준 설명 설정
          object.LevelDescription(i,"Level_"+IntegerToString(i));
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

GetInteger

클래스 인스턴스에 바인딩된 그래픽 개체의 정수 속성 값(bool, bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, 색상 유형)을 수신하기 위해 API MQL5 ObjectGetInteger()의 기능에 대한 단순한 액세스를 제공합니다. 함수 호출에는 두 가지 버전이 있습니다:

정확성을 확인하지 않고 속성 값 가져오기

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // 정수 속성 ID
   int                           modifier=-1       // 제어자
   ) const
```

매개변수

prop_id

[in] 그래픽 개체 double 속성의 ID.

modifier=-1

[in] double 속성의 제어자(인덱스).

값 반환

성공하면 정수 속성 값, 정수 속성을 수신할 수 없으면 0.

작업의 정확성을 확인하는 속성 값 가져오기

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // 정수 속성 ID
   int                           modifier,      // 제어자
   long&                         value          // 변수에 연결
   ) const
```

매개변수

prop_id

[in] 그래픽 개체 정수 속성의 ID.

제어자

[in] 정수 속성의 제어자(인덱스).

값

[out] 정수 속성 값을 배치하려면 변수에 연결.

값 반환

성공하면 true, 정수 속성을 가져오지 못하면 false.

예시:

```
//--- CChartObject::GetInteger 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

© 2000-2025, MetaQuotes Ltd.

```
{
 CChartObject object;
 //--- 쉬운 메서드로 차트 개체 색상 가져오기
 printf("Objects color is %s",ColorToString(object.GetInteger(OBJPROP_COLOR),true));
 //--- 클래식 메서드로 차트 개체 색상 가져오기
 long color_value;
 if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
   {
    printf("정수 속성 오류 %d 가져오기",GetLastError());
    return;
   }
 else
    printf("개체 색상은 %s 입니다",color_value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- 쉬운 메서드로 수준 너비 가져오기
    printf("Level %d width is %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
    //--- 클래식 메서드로 수준 너비 가져오기
    long width_value;
    if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
      {
       printf("정수 속성 오류 %d 가져오기",GetLastError());
       return;
      }
    else
       printf(" %d 수준 너비는 %d 입니다",i,width_value);
   }
}
```

© 2000-2025, MetaQuotes Ltd.

SetInteger

클래스 인스턴스에 바인딩된 그래픽 개체의 정수 속성(of type bool, char, uchar, short, ushort, int, uint, long, ulong, datetime, 색상 유형)을 변경하기 위해 API MQL5 ObjectSetInteger()의 기능에 대한 단순 액세스를 제공합니다. 함수 호출에는 두 가지 버전이 있습니다:

제어자가 필요없는 속성 값 설정

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // 정수 속성 ID
   long                           value         // 값
  \)
```

매개변수

prop_id

[in] 그래픽 개체 정수 속성의 ID.

값

[in] 변경된 정수 속성의 새 값.

제어자를 나타내는 속성 값 설정

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // 정수 속성 ID
   int                            modifier,     // 제어자
   long                           value         // 값
  \)
```

매개변수

prop_id

[in] 그래픽 개체 정수 속성의 ID.

제어자

[in] 정수 속성의 제어자(인덱스).

값

[in] 정수 속성의 새 값.

값 반환

성공하면 true, 정수 속성을 변경하지 못하면 false.

예시:

```
//--- CChartObject::SetInteger 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 새 색상 설정
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!object.SetInteger(OBJPROP_COLOR,clrRed))
     {
      printf("정수 속성 오류 %d 설정하기",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 수준 너비 설정하기
      if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
        {
         printf("정수 속성 에러 %d 설정하기",GetLastError());
         return;
        }
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

GetDouble

API MQL5 ObjectGetDouble()의 기능에 대한 단순 액세스를 제공하여 클래스 인스턴스에 바인딩된 그래픽 개체의 double 값(float 및 double 유형)을 수신. 함수 호출에는 두 가지 버전이 있습니다:

정확성을 확인하지 않고 속성 값 가져오기

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // 정수 속성 ID
   int                          modifier=-1      // 제어자
   ) const
```

매개변수

prop_id

[in] 그래픽 개체 double 속성의 ID.

modifier=-1

[in] double 속성의 제어자(인덱스).

값 반환

성공하면 double 속성 값, double 속성을 수신하지 못하면 EMPTY_VALUE.

이러한 처리의 정확성을 검증할 때 속성 값 가져오기

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,       // double 속성 ID
   int                          modifier,      // 제어자
   double&                      value          // 변수에 연결
   ) const
```

매개변수

prop_id

[in] 그래픽 개체 double 속성의 ID.

제어자

[in] double 속성의 제어자(인덱스).

값

[out] double 속성 값을 배치하려면 변수에 연결.

값 반환

성공하면 true, double 속성을 가져오지 못하면 false.

예시:

```
//--- CChartObject::GetDouble 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
  CChartObject object;
  //---
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- 쉬운 메서드로 수준 값 가져오기
     printf("Level %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
     //--- 클래식 메서드로 수준값 가져오기
     double value;
     if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
       {
        printf("double 속성 오류 %d 가져오기",GetLastError());
        return;
       }
     else
        printf("Level %d value=%f",i,value);
    }
}
```

© 2000-2025, MetaQuotes Ltd.

SetDouble

클래스 인스턴스에 바인딩된 그래픽 개체의 이중 속성(float 및 double 유형)을 변경하기 위해 API MQL5 ObjectSetDouble() 기능에 대한 단순 에액세스를 제공합니다. 함수 호출에는 두 가지 버전이 있습니다:

제어자가 필요없는 속성 값 설정

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // double 속성 ID
   double                        value         // 값
  \)
```

매개변수

prop_id

[in] 그래픽 개체 double 속성의 ID.

값

[in] 변경된 double 속성의 새 값.

제어자를 나타내는 속성 값 설정

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // double 속성 ID
   int                           modifier,     // 제어자
   double                        value         // 값
  \)
```

매개변수

prop_id

[in] 그래픽 개체 double 속성의 ID.

제어자

[in] double 속성의 제어자(인덱스).

값

[in] 변경된 double 속성의 새 값.

값 반환

성공하면 true, double 기능을 변경할 수 없으면 false.

예시:

```
//--- CChartObject::SetDouble 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
//---
   for(int i=0;i<object.LevelsCount();i++)
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 차트 개체의 수준 값 설정
       if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
         {
          printf("doubl e 속성 오류 %d 설정",GetLastError());
          return;
         }
      }
   }
```

© 2000-2025, MetaQuotes Ltd.

GetString

클래스 인스턴스에 바인딩된 그래픽 개체의 문자열 속성 값에 대한 API MQL5 ObjectGetString() 함수에 대한 단순 액세스를 제공. 함수 호출에는 두 가지 버전이 있습니다:

정확성을 확인하지 않고 속성 값 가져오기

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,          // 문자열 속성 ID
   int                          modifier=-1      // 제어자
   ) const
```

매개변수

prop_id

[in] 그래픽 개체 문자열 속성 ID.

modifier=-1

[in] 문자열 속성의 제어자(인덱스).

값 반환

성공하면 문자열 속성 값, 문자열 속성을 수신할 수 없으면 "".

해당 처리의 정확성을 확인하는 속성 값 가져오기

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,       // 문자열 속성 ID
   int                          modifier,      // 제어자
   string&                      value          // 변수에 연결
   ) const
```

매개변수

prop_id

[in] 그래픽 개체 문자열 속성의 ID.

제어자

[in] 문자열 속성의 제어자(인덱스).

값

[out] 문자열 속성 값을 배치하려면 변수에 연결.

값 반환

성공하면 true, 문자열 속성을 가져오지 못하면 false.

예시:

```
//--- CChartObject::GetString 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
   CChartObject object;
   string       value;
   //--- 쉬운 메서로 차트 개체 이름 가져오기
   printf("Object name is '%s'",object.GetString(OBJPROP_NAME));
   //--- 클래식 메서드 차트 개체 이름 가져오기
   if(!object.GetString(OBJPROP_NAME,0,value))
     {
      printf("문자열 속성 오류 %d 가져오기",GetLastError());
      return;
     }
   else
      printf("개체 이름은 '%s' 입니다",value);
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- 쉬운 메서드로 수준 설명 가져오기
      printf(" %d 수준 설명은 '%s' 입니다",i,object.GetString(OBJPROP_LEVELTEXT,i));
      //--- 클래식 메서드 수준 설명 가져오기
      if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
        {
         printf("문자열 속성 오류 %d 가져오기",GetLastError());
         return;
        }
      else
         printf(" %d 수준 설명은 '%s' 입니다",i,value);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

SetString

클래스 인스턴스에 바인딩된 그래픽 개체의 문자열 속성을 변경하기 위한 API MQL5 ObjectSetString() 함수에 대한 단수 액세스를 제공합니다. 함수 호출에는 두 가지 버전이 있습니다:

제어자가 필요없는 속성 값 설정

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // 문자열 속성 ID
   string                        value         // 값
  \)
```

매개변수

prop_id

[in] 그래픽 개체 문자열 속성의 ID.

값

[in] 변경된 문자열 속성의 새 값.

제어자를 나타내는 속성 값 설정

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // 문자열 속성 ID
   int                           modifier,     // 제어자
   string                        value         // 값
  \)
```

매개변수

prop_id

[in] 그래픽 개체 문자열 속성의 ID.

제어자

[in] 문자열 속성의 제어자(인덱스).

값

[in] 변경된 문자열 속성의 새 값.

값 반환

성공하면 true, 문자열 속성을 변경할 수 없으면 false.

예시:

```
//--- CChartObject::SetString 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 차트 개체의 새 이름 설정하기
   if(!object.SetString(OBJPROP_NAME,"MyObject"))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       printf("문자열 속성 오류 %d 설정",GetLastError());
       return;
      }
   for(int i=0;i<object.LevelsCount();i++)
      {
       //--- 수준 설명 설정하기
       if(!object.SetString(OBJPROP_LEVELTEXT,i,"Level_"+IntegerToString(i)))
         {
          printf("문자열 속성 오류 %d 설정",GetLastError());
          return;
         }
      }
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

개체의 매개변수를 파일에 저장하기.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
  \)
```

매개변수

file_handle

[in] FileOpen (...) 기능을 사용하여 이전에 연 파일의 핸들.

값 반환

성공적으로 완료하면 true, 오류이면 false.

예시:

```
//--- CChartObject::Save 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object=new CChartObject;
   //--- 개체 매개변수 설정하기
   //--- . . .
   //--- 파일 열기
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체의 매개변수를 로드.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
  \)
```

매개변수

file_handle

[in] FileOpen (...) 함수를 사용하여 이전에 연 파일의 핸들.

값 반환

성공적으로 완료하면 true, 오류이면 false.

예시:

```
//--- CChartObject::Load 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object;
   //--- 파일 열기
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- 개체 사용
   //--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 ID 가져오기.

```
virtual int  Type() const
```

값 반환

Object type ID (0x8888 for CChartObject).

예시:

```
//--- CChartObject::Type 예시
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- 개체 유형 가져오기
   int type=object.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

라인 객체

"선" 그래픽 개체 그룹.

이 절에는 "선" 그래픽 개체 그룹 작업에 대한 기술 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다.

클래스 이름	Object
CChartObjectVLine	"수직선" 그래픽 개체
CChartObjectHLine	"수평선" 그래픽 개체
CChartObjectTrend	"추세선" 그래픽 개체
CChartObjectTrendByAngle	"각도별 추세선" 그래픽 개체
CChartObjectCycles	"순환선" 그래픽 개체

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

CChartObjectVLine

CChartObjectVLine은 "Vertical Line" 그래픽 객체 속성에 대한 단순화된 액세스를 위한 클래스입니다.

Description

CChartObjectVLine 클래스는 "Vertical Line" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectVLine : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectVLine

그룹별 클래스 메서드

Create	
Create	"Vertical Line" 그래픽 객체를 작성합니다
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"Vertical Line" 그래픽 개체를 만듭니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time           // 시간 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time

[in] 고정점 시간 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_VLINE for CChartObjectVLine).

© 2000-2025, MetaQuotes Ltd.

CChartObjectHLine

CChartObjectHLine은 "Horizontal Line" 그래픽 객체 속성에 대한 단순화된 액세스를 위한 클래스입니다.

Description

CChartObjectHLine 클래스는 "Horizonal Line" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectHLine : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectHLine

그룹별 클래스 메서드

Create	
Create	"Horizontal Line" 그래픽 개체를 만듭니다
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"Horizontal Line" 그래픽 개체를 만듭니다.

```
bool  Create(
   long    chart_id,      // 차트 식별자
   string  name,          // 오브젝트명
   long    window,        // 차트 창
   double  price          // 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

price

[in] 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_HLINE for CChartObjectHLine).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTrend

CChartObjectTrend는 "Trend Line" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectTrend 클래스는 "Trend Line" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectTrend : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend

직계 하위

CChartObjectChannel, CChartObjectFibo, CChartObjectFiboChannel, CChartObjectFiboExpansion, CChartObjectGannFan, CChartObjectGannGrid, CChartObjectPitchfork, CChartObjectRegression, CChartObjectStdDevChannel, CChartObjectTrendByAngle

그룹별 클래스 메서드

Create	
Create	"Trend Line" 그래픽 개체를 만듭니다.
Properties	
RayLeft	"Ray Left" 속성 가져오기/설정
RayRight	"Ray Right" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

더보기

객체 유형, 그래픽 객체

© 2000–2025, MetaQuotes Ltd.

Create

"Trend Line" 그래픽 개체를 만듭니다.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   int       window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 두번째 시간 좌표
   double    price2        // 두번째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

RayLeft (Get Method)

"Ray Left" 속성 값을 가져옵니다.

```
bool  RayLeft() const
```

반환 값

클래스 인스턴스에 할당된 "Ray Left" 속성 값. 할당된 개체가 없으면 false를 반환합니다.

RayLeft (Set Method)

"Ray Left" 속성의 새 플래그 값을 설정합니다.

```
bool  RayLeft(
   bool  ray      // 플래그
   )
```

Parameters

ray

[in] "Ray Left" 속성의 새 값.

반환 값

true – 성공, false – 플래그 변경 불가.

© 2000-2025, MetaQuotes Ltd.

RayRight (Get Method)

"Ray Right" 속성 값을 가져옵니다.

```
bool  RayRight() const
```

반환 값

클래스 인스턴스에 할당된 "Ray Right" 속성 값. 할당된 개체가 없으면 false를 반환합니다.

RayRight (Set Method)

"Ray Right" 속성의 새 플래그 값을 설정합니다.

```
bool  RayRight(
   bool  ray      // 플래그
   )
```

Parameters

ray

[in] "Ray Right" 속성의 새 값.

반환 값

true – 성공, false – 플래그 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이전에 연 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이전에 연 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_TREND for CChartObjectTrend).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTrendByAngle

CChartObjectTrendByAngle은 "Trend Line by Angle" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectTrendByAngle 클래스는 "Trend Line by Angle" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

상속 계층

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle

직계 하위

CChartObjectGannLine

그룹별 클래스 메서드

Create	
Create	"Trend Line by Angle" 그래픽 객체를 작성합니다.
Properties	
Angle	"각도" 속성 가져오기/설정
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000–2025, MetaQuotes Ltd.

Create

"Trend Line by Angle" 그래픽 객체를 만듭니다.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   long      window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 두번째 시간 좌표
   double    price2        // 두번째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Angle (Get Method)

"Angle"도 속성 값을 가져옵니다.

```
double  Angle() const
```

반환 값

클래스 인스턴스에 할당된 "Angle" 속성의 값. 할당된 객체가 없으면 EMPTY_VALUE를 반환합니다.

Angle (Set Method)

"Angle" 속성의 새 값을 설정합니다.

```
bool  Angle(
   double  angle      // 각도
   )
```

Parameters

angle

[in] "Angle" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_TRENDBYANGLE for CChartObjectTrendByAngle).

© 2000-2025, MetaQuotes Ltd.

CChartObjectCycles

CChartObjectCycles는 "Cyclic Lines" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectCycles는 "Cyclic Lines" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectCycles : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectCycles

그룹별 클래스 메서드

Create	
Create	"Cycle Lines" 그래픽 개체를 만듭니다.
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"Cyclic Lines" 그래픽 개체를 만듭니다.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   long      window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 두번째 시간 좌표
   double    price2        // 두번째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_CYCLES for CChartObjectCycles).

© 2000-2025, MetaQuotes Ltd.

채널 객체

"채널" 그래픽 개체 그룹.

이 절에는 "Channels" 그래픽 개체 그룹 작업에 대한 기술적 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다.

클래스 이름	Object
CChartObjectChannel	"등거리 채널" 그래픽 개체
CChartObjectRegression	"선형 회귀 채널" 그래픽 객체
CChartObjectStdDevChannel	"표준 편차 채널" 그래픽 객체
CChartObjectPitchfork	"앤드류 피치포크" 그래픽 객체

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

CChartObjectChannel

CChartObjectChannel は "등거리 채널" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectChannel 클래스는 "등거리 채널" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectChannel : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectChannel

그룹별 클래스 메서드

Create	
Create	"등거리 채널" 그래픽 객체를 생성합니다
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"등거리 채널" 그래픽 개체를 만듭니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 고정 포인트 시간 좌표
   double    price1,        // 첫 고정 포인트 가격 좌표
   datetime  time2,         // 두번째 고정점 시간 좌표
   double    price2,        // 두번째 고정점 가격 좌표
   datetime  time3,         // 세번째 고정점 시간 좌표
   double    price3         // 세번째 고정점 가격 좌표
   )
```

Parameters

chart_id

[in] Chart ID (0 - 현 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

time3

[in] 세번째 고정점 시간 좌표.

price3

[in] 세번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_CHANNEL for CChartObjectChannel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectRegression

CChartObjectRegression 은 "선형 회귀 채널" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다 .

Description

CChartObjectRegression 클래스는 "선형 회귀 채널" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectRegression : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectRegression

그룹별 클래스 메서드

Create	
Create	"선형 회귀 채널" 그래픽 개체를 생성합니다
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

더보기

© 2000-2025, MetaQuotes Ltd.

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

선형 회귀 채널 그래픽 개체를 생성합니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   long      window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   datetime  time2          // 둘째 시간 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 – 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 – 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

time2

[in] 두번째 고정점 시간 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_REGRESSION for CChartObjectRegression).

© 2000-2025, MetaQuotes Ltd.

CChartObjectStdDevChannel

CChartObjectStdDevChannel은 "표준 편차 채널" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectStdDevChannel 클래스는 "표준 편차 채널" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectStdDevChannel

그룹별 클래스 메서드

Create	
Create	"표준 편차 채널" 그래픽 객체를 생성합니다
Properties	
Deviations	"편차" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"표준 편차 채널" 그래픽 객체 생성.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    deviation      // 편차
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

time2

[in] 두번째 고정점 시간 좌표.

deviation

[in] "편차" 속성의 숫자 값.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Deviation (Get Method)

"편차" 속성 값을 가져옵니다.

```
double  Deviation() const
```

반환 값

클래스 인스턴스에 할당된 "편차" 속성의 값입니다. 할당된 객체가 없으면 EMPTY_VALUE를 반환합니다.

편차 (메서드 설정)

"편차" 속성 값을 설정합니다.

```
bool  Deviation(
   double  deviation      // 편차
   )
```

Parameters

deviation

[in] "편차" 속성용 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 속성 식별자 (OBJ_STDDEVCHANNEL for CChartObjectStdDevChannel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectPitchfork

CChartObjectPitchfork는 "앤드류 피치포크" 그래픽 객체 속성에 대한 단순화된 액세스를 위한 클래스입니다.

Description

CChartObjectPitchfork 클래스는 "앤드류 피치포크" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectPitchfork

그룹별 클래스 메서드

Create	
Create	"앤드류 피치포크" 그래픽 객체 생성
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

더보기

© 2000-2025, MetaQuotes Ltd.

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"앤드류 피치포크" 그래픽 객체 생성.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   long      window,       // 차트 창
   datetime  time1,        // 첫 고정 포인트 시간 좌표
   double    price1,       // 첫 고정 포인트 가격 좌표
   datetime  time2,        // 두번째 고정점 시간 좌표
   double    price2,       // 두번째 고정점 가격 좌표
   datetime  time3,        // 세번째 고정점 시간 좌표
   double    price3        // 세번째 고정점 가격 좌표
   )
```

Parameters

chart_id
  [in] 차트 식별자 (0 - 현재 차트).

name
  [in] 생성할 객체의 고유 이름.

window
  [in] 차트 창 번호 (0 - 메인 창).

time1
  [in] 첫 고정점의 시간 좌표.

price1
  [in] 첫 고정점 가격 좌표.

time2
  [in] 두번째 고정점 시간 좌표.

price2
  [in] 두번째 고정점 가격 좌표.

time3
  [in] 세번째 고정점 시간 좌표.

price3
  [in] 세번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_PITCHFORK for CChartObjectPitchfork).

© 2000-2025, MetaQuotes Ltd.

간(Gann) 툴

그룹 "간 툴" 그래픽 개체.

이 절에는 "간 툴" 그래픽 개체 그룹 작업에 대한 기술적 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 수록되어 있다.

클래스 이름	Object
CChartObjectGannLine	"간 선" 그래픽 객체
CChartObjectGannFan	"간 팬" 그래픽 개체
CChartObjectGannGrid	"간 그리드" 그래픽 개체

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannLine

CChartObjectGannLine은 "간 선" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectGannLine 클래스는 "간 선" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

상속 계층

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle
CChartObjectGannLine

그룹별 클래스 메서드

Create	
Create	"간(Gann) 선" 그래픽 객체 생성
Properties	
PipsPerBar	"막대당 Pips" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create

CChartObjectTrendByAngle 클래스에서 상속된 메서드

Angle, Angle, Create

더보기

객체 유형, 그래픽 객체

© 2000–2025, MetaQuotes Ltd.

Create

"간(Gann) 선" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   double    price1,        // 첫 가격 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    ppb            // 막대당 pips
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

ppb

[in] 막대당 Pips.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (Get Method)

"막대당 Pips" 속성 값을 가져옵니다.

```
double  PipsPerBar() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "막대당 Pips" 속성 값입니다. 할당된 개체가 없으면 EMPTY_VALUE를 반환합니다.

PipsPerBar (Set Method)

"막대당 Pips" 속성의 새 값을 설정합니다.

```
bool  PipsPerBar(
   double  ppb      // 막대당 pips
   )
```

Parameters

ppb

[in] "막대당 Pips" 속성용 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_GANNLINE for CChartObjectGannLine).

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannFan

CChartObjectGannFan은 "간 팬" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectGannFan 클래스는 "간 팬" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectGannFan : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannFan

그룹별 클래스 메서드

Create	
Create	"간 팬" 그래픽 객체를 생성합니다
Properties	
PipsPerBar	"막대당 Pips" 속성 가져오기/설정
Downtrend	"다운트렌드" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create

더보기

객체 유형, 그래픽 객체

© 2000–2025, MetaQuotes Ltd.

Create

"간 팬" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   double    price1,        // 첫 가격 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    ppb            // 막대당 pi ps
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 기본 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

ppb

[in] 막대 당 pip.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (Get Method)

막대당 Pips 속성 값을 가져옵니다.

```
double  PipsPerBar() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "막대당 Pips" 속성 값입니다. 할당된 객체가 없으면 EMPTY_VALUE를 반환합니다.

PipsPerBar (Set Method)

"막대당 Pips" 속성의 새 값을 설정합니다.

```
bool  PipsPerBar(
   double  ppb      // 막대당 pips
   )
```

Parameters

ppb

[in] 막대당 Pips 속성에 대한 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Downtrend (Get Method)

"Downtrend" 플래그 값을 가져옵니다.

```
bool  Downtrend() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Downtrend" 플래그 값. 할당된 개체가 없으면 false를 반환합니다.

Downtrend (Set Method)

"Downtrend" 속성의 새 값을 설정합니다.

```
bool  Downtrend(
   bool  downtrend      // 플래그 값
   )
```

Parameters

downtrend

[in] "다운트렌드" 속성의 새 값.

반환 값

true - 성공, false - 플래그 변경 실패.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_GANNFAN for CChartObjectGannFan).

© 2000-2025, MetaQuotes Ltd.

CChartObjectGannGrid

CChartObjectGannGrid는 "간 그리드" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectGannGrid 클래스는 "간 그리드" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannGrid

그룹별 클래스 메서드

Create	
Create	"간 그리드" 그래픽 객체를 생성합니다
Properties	
PipsPerBar	"막대당 Pips" 속성 가져오기/설정
Downtrend	"다운트렌드" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create

더보기

객체 유형, 그래픽 객체

© 2000–2025, MetaQuotes Ltd.

Create

"간 그리드" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   double    price1,        // 첫 가격 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    ppb            // 막대당 pi ps
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

ppb

[in] 막대 당 pip.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

PipsPerBar (Get Method)

막대당 Pips 속성 값을 가져옵니다.

```
double  PipsPerBar() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "막대당 Pips" 속성 값입니다. 할당된 객체가 없으면 EMPTY_VALUE를 반환합니다.

PipsPerBar (Set Method)

"막대당 Pips" 속성의 새 값을 설정합니다.

```
bool  PipsPerBar(
   double  ppb      // 막대당 Pips
   )
```

Parameters

ppb

[in] 막대당 Pips 속성에 대한 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Downtrend (Get Method)

"다운트렌드" 속성 값을 가져옵니다.

```
bool  Downtrend() const
```

반환 값

클래스 인스턴트에 할당된 객체의 "다운트렌드" 속성 값입니다. 할당된 개체가 없으면 false를 반환합니다.

Downtrend (Set Method)

"Downtrend" 속성의 새 값을 설정합니다.

```
bool  Downtrend(
   bool  downtrend      // 플래그 값
   )
```

Parameters

downtrend

[in] "다운트렌드" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_GANNGRID for CChartObjectGannGrid).

© 2000-2025, MetaQuotes Ltd.

피보나치 툴

"Fibonacci Tools" 그래픽 개체 그룹.

이 절에는 "Fibonacci Tools" 그래픽 개체 그룹 작업에 대한 기술 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다.

클래스 이름	Object
CChartObjectFibo	"피보나치 되돌림" 그래픽 객체
CChartObjectFiboTimes	"피보나치 시간대" 그래픽 객체
CChartObjectFiboFan	"피보나치 팬" 그래픽 객체
CChartObjectFiboArc	"피보나치 호" 그래픽 객체
CChartObjectFiboChannel	"피보나치 채널" 그래픽 객체
CChartObjectFiboExpansion	"피보나치 확장" 그래픽 객체

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

CChartObjectFibo

CChartObjectFibo는 "피보나치 되돌림" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectFibo 클래스는 "피보나치 되돌림" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectFibo : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFibo

그룹별 클래스 메서드

Create	
Create	"피보나치 되돌림" 그래픽 객체 생성
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"피보나치 되돌림" 그래픽 객체 생성.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   int       window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 둘째 시간 좌표
   double    price2        // 둘째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_FIBO for CChartObjectFibo).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboTimes

CChartObjectFiboTimes는 "피보나치 시간대" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectFiboTimes 클래스는 "피보나치 시간대" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectFiboTimes : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectFiboTimes

그룹별 클래스 메서드

Create	
Create	"피보나치 시간대" 그래픽 객체를 생성합니다
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"피보나치 시간대" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   int       window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 둘째 시간 좌표
   double    price2        // 둘째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_FIBOTIMES for CChartObjectFiboTimes).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboFan

CChartObjectFiboFan은 "피보나치 팬" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectFiboFan 클래스는 "피보나치 팬" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectFiboFan : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectFiboFan

그룹별 클래스 메서드

Create	
Create	"피보나치 팬" 그래픽 객체
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"피보나치 팬" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   int       window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 둘째 시간 좌표
   double    price2        // 둘째 가격 좌표
   )
```

Parameters

chart_id
[in] 차트 식별자 (0 – 현재 차트).

name
[in] 생성할 개체의 고유 이름.

window
[in] 차트 창 번호 (0 – 메인 창).

time1
[in] 첫 고정점의 시간 좌표.

price1
[in] 첫 고정점 가격 좌표.

time2
[in] 두번째 고정점 시간 좌표.

price2
[in] 두번째 고정점 가격 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_FIBOFAN for CChartObjectFiboFan).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboArc

CChartObjectFiboArc는 "피보나치 호" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectFiboArc 클래스는 "피보나치 호" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectFiboArc : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectFiboArc

그룹별 클래스 메서드

Create	
Create	"피보나치 호" 그래픽 객체 생성
Properties	
Scale	"스케일" 속성 가져오기/설정
Ellipse	"타원" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

© 2000-2025, MetaQuotes Ltd.

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"피보나치 호" 그래픽 객체 생성.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   int       window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 둘째 시간 좌표
   double    price2,       // 둘째 가격 좌표
   double    scale         // 스케일
   )
```

Parameters

chart_id
  [in] 차트 식별자 (0 - 현재 차트).

name
  [in] 생성할 개체의 고유 이름.

window
  [in] 차트 창 번호 (0 - 메인 창).

time1
  [in] 첫 고정점의 시간 좌표.

price1
  [in] 첫 고정점 가격 좌표.

time2
  [in] 두번째 고정점 시간 좌표.

price2
  [in] 두번째 고정점 가격 좌표.

scale
  [in] 스케일.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Scale (Get Method)

"스케일" 속성 값을 가져옵니다.

```
double  Scale() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "스케일" 속성 값. 할당된 객체가 없으면 EMPTY_VALUE를 반환합니다.

Scale (Set Method)

"스케일" 속성용 새 값을 설정합니다.

```
bool  Scale(
   double  scale      // 스케일
   )
```

Parameters

scale

[in] "스케일" 속성용 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Ellipse (Get Method)

"Ellipse" 플래그 값을 가져옵니다.

```
bool  Ellipse() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Ellipse" 플래그 값입니다. 할당된 개체가 없으면 false를 반환합니다.

Ellipse (Set Method)

"Ellipse" 플래그 값을 설정합니다.

```
bool  Ellipse(
   bool  ellipse      // 플래그 값
   )
```

Parameters

ellipse

[in] "타원" 속성용 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_FIBOARC for CChartObjectFiboArc).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboChannel

CChartObjectFiboChannel은 "피보나치 채널" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectFiboChannel 클래스는 "피보나치 채널" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

상속 계층

CObject
CChartObject
CChartObjectTrend
CChartObjectFiboChannel

그룹별 클래스 메서드

Create	
Create	"피보나치 채널" 그래픽 객체
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

더보기

© 2000-2025, MetaQuotes Ltd.

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"피보나치 채널" 그래픽 객체.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   double    price1,        // 첫 가격 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    price2,        // 둘째 가격 좌표
   datetime  time3,         // 셋째 시간 좌표
   double    price3         // 셋째 가격 좌표
   )
```

Parameters

chart_id
  [in] 차트 식별자 (0 – 현재 차트).

name
  [in] 생성할 개체의 고유 이름.

window
  [in] 차트 창 번호 (0 – 메인 창).

time1
  [in] 첫 고정점의 시간 좌표.

price1
  [in] 첫 고정점 가격 좌표.

time2
  [in] 두번째 고정점 시간 좌표.

price2
  [in] 두번째 고정점 가격 좌표.

time3
  [in] 세번째 고정점 시간 좌표.

price3
  [in] 세번째 고정점 가격 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_FIBOCHANNEL for CChartObjectFiboChannel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectFiboExpansion

CChartObjectFiboExpansion은 "피보나치 확장" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectFiboExpansion 클래스는 "피보나치 확장" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboExpansion

그룹별 클래스 메서드

Create	
Create	"피보나치 확장" 그래픽 객체 생성
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectTrend 클래스로부터 상속된 메서드

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

더보기

© 2000-2025, MetaQuotes Ltd.

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"피보나치 확장" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   double    price1,        // 첫 가격 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    price2,        // 둘째 가격 좌표
   datetime  time3,         // 셋째 시간 좌표
   double    price3         // 셋째 가격 좌표
   )
```

Parameters

chart_id
  [in] 차트 식별자 (0 - 현재 차트).

name
  [in] 생성할 개체의 고유 이름.

window
  [in] 차트 창 번호 (0 - 메인 창).

time1
  [in] 첫 고정점의 시간 좌표.

price1
  [in] 첫 고정점 가격 좌표.

time2
  [in] 두번째 고정점 시간 좌표.

price2
  [in] 두번째 고정점 가격 좌표.

time3
  [in] 세번째 고정점 시간 좌표.

price3
  [in] 세번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_EXPANSION for CChartObjectFiboExpansion).

© 2000-2025, MetaQuotes Ltd.

엘리엇 툴

"Elliott Tools" 그래픽 개체 그룹.

이 절에는 "Elliott Tools" 그래픽 개체 클래스 그룹에 대한 자세한 내용이 포함되어 있습니다.

클래스 이름	Object
CChartObjectElliottWave3	"조정 파동" 그래픽 객체
CChartObjectElliottWave5	"추진 파동" 그래픽 객체

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

CChartObjectElliottWave3

CChartObjectElliottWave3 "조정 파동" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectElliottWave3 클래스는 "조정 파동" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectElliottWave3 : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectElliottWave3

직계 하위

CChartObjectElliottWave5

그룹별 클래스 메서드

Create	
Create	"조정 파동" 그래픽 객체 생성
Properties	
Degree	"각도" 속성 가져오기/설정
Lines	"선" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

더보기

객체 유형, 그래픽 객체

© 2000–2025, MetaQuotes Ltd.

Create

"조정 파동" 그래픽 객체 생성.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   double    price1,        // 첫 가격 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    price2,        // 둘째 가격 좌표
   datetime  time3,         // 셋째 시간 좌표
   double    price3         // 셋째 가격 좌표
   )
```

Parameters

chart_id
  [in] 차트 식별자 (0 – 현재 차트).

name
  [in] 생성할 개체의 고유 이름.

window
  [in] 차트 창 번호 (0 – 메인 창).

time1
  [in] 첫 고정점의 시간 좌표.

price1
  [in] 첫 고정점 가격 좌표.

time2
  [in] 두번째 고정점 시간 좌표.

price2
  [in] 두번째 고정점 가격 좌표.

time3
  [in] 세번째 고정점 시간 좌표.

price3
  [in] 세번째 고정점 시간 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

Degree (Get Method)

"각도: 속성 값 가져오기

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "각도" 속성 값. 할당된 개체가 없으면 WORK_VALUE 를 반환합니다.

Degree (Set Method)

"각도" 속성의 새 값을 설정합니다.

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // 속성 값
   )
```

Parameters

degree

[in] "각도" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Lines (Get Method)

"선" 속성 값을 가져옵니다.

```
bool  Lines() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "선" 속성 값. 할당된 개체가 없으면 false를 반환합니다.

Lines (Set Method)

"선" 속성의 새 값을 설정합니다.

```
bool  Lines(
   bool  lines      // 플래그 값
   )
```

Parameters

lines

[in] "선" 속성의 새 값.

반환 값

true - 성공, false - 플래그 변경 실패.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_ELLIOTWAVE3 for CChartObjectElliottWave3).

© 2000-2025, MetaQuotes Ltd.

CChartObjectElliottWave5

CChartObjectElliottWave5는 "추진 파동" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectElliottWave5 클래스는 "추진 파동" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Title

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectElliottWave3
      CChartObjectElliottWave5

그룹별 클래스 메서드

Create	
Create	"추진 파동" 그래픽 객체 생성
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

클래스 CChartObjectElliottWave3에서 상속된 메서드

Degree, Degree, Lines, Lines, Create, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"추진 파동" 그래픽 객체 생성.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   double    price1,        // 첫 가격 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    price2,        // 둘째 가격 좌표
   datetime  time3,         // 셋째 시간 좌표
   double    price3,        // 셋째 가격 좌표
   datetime  time4,         // 넷째 시간 좌표
   double    price4,        // 넷째 가격 좌표
   datetime  time5          // 다섯째 시간 좌표
   double    price5,        // 다섯째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 – 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 – 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

time3

[in] 세번째 고정점 시간 좌표.

price3

[in] 세번째 고정점 가격 좌표.

time4

[in] 네 번째 고정점의 시간 좌표.

© 2000-2025, MetaQuotes Ltd.

price4

[in] 네 번째 고정점의 가격 좌표.

time5

[in] 다섯 번째 고정점의 시간 좌표.

price5

[in] 다섯 번째 기준점의 가격 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_ELLIOTWAVE5 for CChartObjectElliottWave5).

© 2000-2025, MetaQuotes Ltd.

도형 객체

"도형" 그래픽 개체 그룹.

이 절에는 "도형" 그래픽 개체 그룹 작업에 대한 기술 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다.

클래스 이름	Object
CChartObjectRectangle	"사각형" 그래픽 객체
CChartObjectTriangle	"삼각형" 그래픽 객체
CChartObjectEllipse	"타원" 그래픽 객체

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

CChartObjectRectangle

CChartObjectRectangle은 "사각형" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectRectangle 클래스는 "사각형" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectRectangle : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectRectangle

그룹별 클래스 메서드

Create	
Create	"사각형" 사각형 객체 생성
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"사각형" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   long      window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 둘째 시간 좌표
   double    price2        // 둘째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_RECTANGLE for CChartObjectRectangle).

© 2000-2025, MetaQuotes Ltd.

CChartObjectTriangle

CChartObjectTriangle은 "삼각형" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectTriangle 클래스는 "삼각형" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectTriangle : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectTriangle

그룹별 클래스 메서드

Create	
Create	"삼각형" 그래픽 객체를 생성합니다
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"삼각형" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   long      window,        // 차트 창
   datetime  time1,         // 첫 시간 좌표
   double    price1,        // 첫 가격 좌표
   datetime  time2,         // 둘째 시간 좌표
   double    price2,        // 둘째 가격 좌표
   datetime  time3,         // 셋째 시간 좌표
   double    price3         // 셋째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 – 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 – 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

time3

[in] 세번째 고정점 시간 좌표.

price3

[in] 세번째 고정점 가격 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_TRIANGLE for CChartObjectTriangle).

© 2000-2025, MetaQuotes Ltd.

CChartObjectEllipse

CChartObjectEllipse는 "타원" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스.

Description

CChartObjectEllipse 클래스는 "타원" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectEllipse : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectEllipse

그룹별 클래스 메서드

Create	
Create	"타원" 그래픽 객체 생성
입/출력	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint, Save, Load

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"타원" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,     // 차트 식별자
   string    name,         // 객체명
   int       window,       // 차트 창
   datetime  time1,        // 첫 시간 좌표
   double    price1,       // 첫 가격 좌표
   datetime  time2,        // 둘째 시간 좌표
   double    price2,       // 둘째 가격 좌표
   datetime  time3,        // 셋째 시간 좌표
   double    price3        // 셋째 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 – 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 – 메인 창).

time1

[in] 첫 고정점의 시간 좌표.

price1

[in] 첫 고정점 가격 좌표.

time2

[in] 두번째 고정점 시간 좌표.

price2

[in] 두번째 고정점 가격 좌표.

time3

[in] 세번째 고정점 시간 좌표.

price3

[in] 세번째 고정점 가격 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_ELLIPSE for CChartObjectEllipse).

© 2000-2025, MetaQuotes Ltd.

화살표 객체

화살표용 그룹 그래픽 객체.

이 절에는 "화살표" 그래픽 개체 그룹 작업에 대한 기술 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다. 본질적으로 화살표는 특정 코드와 연결된 특정 아이콘입니다. 차트에 아이콘을 표시하는 "화살표" 그래픽 개체에는 두 가지 유형이 있습니다.

- "화살표" 개체 –개체가 표시하는 아이콘의 코드를 지정할 수 있습니다.
- 특정 유형의 아이콘(특정 고정 코드에 해당)을 표시할 개체 그룹입니다.

임의 코드 아이콘을 표시하는 화살표 작업을 위한 클래스

클래스 이름	화살표 개체의 이름
CChartObjectArrow	Arrow

고정 코드 아이콘을 표시하는 작업 화살표 클래스

클래스 이름	화살표 개체의 이름
CChartObjectArrowCheck	Check
CChartObjectArrowDown	위쪽 화살표
CChartObjectArrowUp	아래쪽 화살표
CChartObjectArrowStop	정지 신호
CChartObjectArrowThumbDown	추천
CChartObjectArrowThumbUp	비추천
CChartObjectArrowLeftPrice	Left Price Label
CChartObjectArrowRightPrice	Right Price Label

더보기

객체 유형, 객체 바인딩 메서드, 그래픽 객체

© 2000–2025, MetaQuotes Ltd.

CChartObjectArrow

CChartObjectArrow는 "화살표" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectArrow 클래스는 모든 하위 항목에 대한 "화살표" 객체의 공통 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectArrow : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

상속 계층

CObject
CChartObject
CChartObjectArrow

직계 하위

CChartObjectArrowCheck, CChartObjectArrowDown, CChartObjectArrowLeftPrice, CChartObjectArrowRightPrice, CChartObjectArrowStop, CChartObjectArrowThumbDown, CChartObjectArrowThumbUp, CChartObjectArrowUp

그룹별 클래스 메서드

Create	
Create	생성: "화살표" 그래픽 오브젝트
Properties	
ArrowCode	"화살표 코드" 속성 가져오기/설정
Anchor	"고정" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

더보기

객체 유형, 객체 바인딩 메서드, 그래픽객체

© 2000–2025, MetaQuotes Ltd.

Create

Creates "Arrow" 그래픽 객체.

```
bool  Create(
   long      chart_id,      // 차트 ID
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time,          // 시간
   double    price,         // 가격
   char      code           // 화살표 코드
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 고유 객체명.

window

[in] 차트 창 번호 (0 - 메인 창).

time

[in] 시간 좌표.

price

[in] 값 좌표.

code

[in] "화살표" 코드 (Wingdings).

반환 값

true - 성공, false - 오류.

예제:

```
//--- CChartObjectArrow::Create 의 예
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
//--- 객체 매개 변수 설정
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- 화살표 생성 오류
      printf("화살표 생성: 오류 %d!",GetLastError());
      //---
```

© 2000-2025, MetaQuotes Ltd.

```
      return;
     }
//--- 화살표 사용
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrowCode (Get Method)

"화살표" 심볼 코드를 가져옵니다.

```
char  ArrowCode() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "화살표" 심볼 코드입니다. 할당된 객체가 없으면 0을 반환합니다.

ArrowCode (메서드 설정)

"화살표"의 심볼 코드를 설정합니다

```
bool  ArrowCode(
   char  code      // 코드 값
   )
```

Parameters

code

[in] "화살표" 코드(Wingdings)의 새 값.

반환 값

true – 성공, false – 코드 변경 불가.

예제:

```
//--- CChartObjectArrow::ArrowCode 의 예
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   char              code=181;
//--- 객체 매개 변수 설정
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
     {
      //--- 화살표 생성 오류
      printf("화살표 생성: 오류 %d!",GetLastError());
      //---
      return;
     }
//--- 화살표 코드 변경
//--- . . .
//--- 화살표 코드 가져오기
   if(arrow.ArrowCode()!=code)
     {
     //--- 화살표 코드 설정
     arrow.ArrowCode(code);
```

© 2000-2025, MetaQuotes Ltd.

```
      }
//--- 화살표 사용
//--- . . .
   }
```

© 2000-2025, MetaQuotes Ltd.

Anchor (Get Method)

화살표 개체의 고정 유형을 가져옵니다.

```
ENUM_ARROW_ANCHOR  Anchor() const
```

반환 값

차트(클래스 인스턴스)에 할당된 "화살표" 개체의 고정 유형. 할당된 개체가 없으면 WORK_VALUE 를 반환합니다.

Anchor (Set Method)

화살표 개체의 앵커(고정) 유형을 설정합니다.

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // 고정 유형
   )
```

Parameters

anchor

[in] 새 고정 유형 값

반환 값

true - 성공, false - 고정 유형 변경 실패.

예제:

```
//--- CChartObject::Anchor 의 예
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
//--- 객체 매개 변수 설정
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- 화살표 생성 오류
      printf("화살표 생성: 오류 %d!",GetLastError());
      //---
      return;
     }
//--- 화살표 고정 가져오기
   if(arrow.Anchor()!=anchor)
     {
     //--- 화살표 고정 생성
     arrow.Anchor(anchor);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 화살표 사용
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] 이전에 FileOpen(...) 함수를 사용하여 연 파일의 핸들.

반환 값

true - 성공, false - 오류.

예제:

```
//--- CChartObjectArrow::Save 의 예
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- 객체 매개 변수 설정
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- 화살표 생성 오류
      printf("화살표 생성: 오류 %d!",GetLastError());
      //---
      return;
     }
//--- 파일 열기
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Save(file_handle))
        {
         //--- 파일 저장 오류
         printf("파일 저장: 오류 %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen(...) 함수를 사용하여 이전에 열었던 파일의 핸들.

반환 값

true - 성공, false - 오류.

예제:

```
//--- CChartObjectArrow::Load 의 예
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int                file_handle;
   CChartObjectArrow arrow;
//--- 파일 열기
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Load(file_handle))
        {
         //--- 파일 로드 오류
         printf("파일 로드: 오류 %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
//--- 화살표 사용
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

Object type identifier (for example, OBJ_ARROW for CChartObjectArrow)

예제:

```
//--- CChartObjectArrow::Type 의 예
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- 화살표 유형 가져오기
   int type=arrow.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

고정 코드가 있는 화살표

"고정 코드가 있는 화살표"는 다음 그래픽 개체의 속성에 대한 단순 액세스를 위한 클래스입니다.

클래스 이름	화살표 객체명
CChartObjectArrowCheck	"체크 화살표"
CChartObjectArrowDown	"아래쪽 화살표"
CChartObjectArrowUp	"위쪽 화살표"
CChartObjectArrowStop	"정지 화살표"
CChartObjectArrowThumbDown	"좋아요" ("추천")
CChartObjectArrowThumbUp	"싫어요" ("비추천")
CChartObjectArrowLeftPrice	"좌측 가격" 화살표
CChartObjectArrowRightPrice	"우측 가격" 화살표

Description

"고정 코드가 있는 화살표" 클래스는 개체 속성에 대한 액세스를 제공합니다.

Declarations

```
class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown       : public CChartObjectArrow;
class CChartObjectArrowUp         : public CChartObjectArrow;
class CChartObjectArrowStop       : public CChartObjectArrow;
class CChartObjectArrowThumbDown  : public CChartObjectArrow;
class CChartObjectArrowThumbUp    : public CChartObjectArrow;
class CChartObjectArrowLeftPrice  : public CChartObjectArrow;
class CChartObjectArrowRightPrice : public CChartObjectArrow;
```

Title

```
<ChartObjects\ChartObjectsArrows.mqh>
```

그룹별 클래스 메서드

Create	
Create	클래스와 일치하는 그래픽 개체를 만듭니다
Properties	
ArrowCode	심볼 코드 변경 방법에 대한 "Stub"
입/출력	
virtual Type	가상식별방법

© 2000-2025, MetaQuotes Ltd.

더보기

객체 유형, 객체 바인딩 메서드, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"고정 코드가 있는 화살표" 그래픽 객체 생성.

```
bool  Create(
   long      chart_id,      // 차트 ID
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time,          // 시간
   double    price          // 가격
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 객체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

time

[in] 시간 좌표.

price

[in] 값 좌표.

반환 값

true - 성공, false - 오류.

예제:

```
//--- CChartObjectArrowCheck::Create 의 예
//--- CChartObjectArrowDown::Create 의 예
//--- CChartObjectArrowUp::Create 의 예
//--- CChartObjectArrowStop::Create 의 예
//--- CChartObjectArrowThumbDown::Create 의 예
//--- CChartObjectArrowThumbUp::Create 의 예
//--- CChartObjectArrowLeftPrice::Create 의 예
//--- CChartObjectArrowRightPrice::Create 의 예
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- 예를 들어, CChartObjectArrowCheck를 택합니다
  CChartObjectArrowCheck 화살표;
//--- 객체 매개 변수 설정
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

© 2000-2025, MetaQuotes Ltd.

```
      {
       //--- 화살표가 오류 생성
       printf("화살표 생성: 오류 %d!",GetLastError());
       //---
       return;
      }
//--- 화살표 사용
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

ArrowCode

화살표 심볼 코드를 변경할 수 없습니다.

```
bool  ArrowCode(
   char  code      // 코드 값
   )
```

Parameters

code

[in] 아무 값

반환 값

항상 false.

예제:

```
//--- example for CChartObjectArrowCheck::ArrowCode
//--- example for CChartObjectArrowDown::ArrowCode
//--- example for CChartObjectArrowUp::ArrowCode
//--- example for CChartObjectArrowStop::ArrowCode
//--- example for CChartObjectArrowThumbDown::ArrowCode
//--- example for CChartObjectArrowThumbUp::ArrowCode
//--- example for CChartObjectArrowLeftPrice::ArrowCode
//--- example for CChartObjectArrowRightPrice::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- 예를 들어, CChartObjectArrowCheck를 택합니다
  CChartObjectArrowCheck 화살표;
//--- 객체 매개 변수 설정
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
     {
      //--- 화살표 생성 오류
      printf("Arrow create: Error %d!",GetLastError());
      //---
      return;
     }
//--- 화살표 코드 설정
   if(!arrow.ArrowCode(181))
     {
      //--- 오류가 아닙니다
      printf("화살표 코드는 변경할 수 없습니다");
     }
//--- 화살표 사용
//--- . . .
  }
```

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다

```
virtual int  Type() const
```

반환 값

객체 유형 식별자:

CChartObjectArrowCheck - OBJ_ARROW_CHECK,

CChartObjectArrowDown - OBJ_ARROW_DOWN,

CChartObjectArrowUp - OBJ_ARROW_UP,

CChartObjectArrowStop - OBJ_ARROW_STOP,

CChartObjectArrowThumbDown - OBJ_ARROW_THUMB_DOWN,

CChartObjectArrowThumbUp - OBJ_ARROW_THUMB_UP,

CChartObjectArrowLeftPrice - OBJ_ARROW_LEFT_PRICE,

CChartObjectArrowRightPrice - OBJ_ARROW_RIGHT_PRICE.

예제:

```
//--- example for CChartObjectArrowCheck::Type
//--- example for CChartObjectArrowDown::Type
//--- example for CChartObjectArrowUp::Type
//--- example for CChartObjectArrowStop::Type
//--- example for CChartObjectArrowThumbDown::Type
//--- example for CChartObjectArrowThumbUp::Type
//--- example for CChartObjectArrowLeftPrice::Type
//--- example for CChartObjectArrowRightPrice::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- 예를 들어, CChartObjectArrowCheck를 택합니다
  CChartObjectArrowCheck 화살표;
//--- 화살표 유형 가져오기
   int type=arrow.Type();
  }
```

© 2000-2025, MetaQuotes Ltd.

제어 객체

"Object Controls" 그래픽 개체 그룹.

이 절에는 "Object Controls" 그래픽 개체 그룹 작업에 대한 기술 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다.

클래스 이름	Object
CChartObjectText	"텍스트" 그래픽 객체
CChartObjectLabel	"텍스트 레이블" 그래픽 객체
CChartObjectEdit	"편집" 그래픽 객체
CChartObjectButton	"버튼" 그래픽 객체
CChartObjectSubChart	"차트" 그래픽 객체
CChartObjectBitmap	"비트맵" 그래픽 객체
CChartObjectBmpLabel	"비트맵 레이블" 그래픽 객체
CChartObjectRectLabel	"정사각 레이블" 그래픽 객체

더보기

객체 유형, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

CChartObjectText

CChartObjectText는 "Text" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectText 클래스는 "Text" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectText : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

상속 계층

CObject
CChartObject
CChartObjectText

직계 하위

CChartObjectLabel

그룹별 클래스 메서드

Create	
Create	"Text" 그래픽 객체를 생성합니다
Properties	
Angle	"각도" 속성 가져오기/설정
Font	"Font" 속성 가져오기/설정
FontSize	"FontSize" 속성 가져오기/설정
Anchor	"고정" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Derived classes:

- CChartObjectLabel

더보기

객체 유형, 객체 속성, 객체 바인딩 메서드, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"Text" 그래픽 개체를 만듭니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time,          // 시간 좌표
   double    price          // 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 – 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 – 메인 창).

time

[in] 고정점 시간 좌표.

price

[in] 고정점 가격 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

Angle (Get Method)

"Angle"도 속성 값을 가져옵니다.

```
double  Angle() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Angle" 속성 값. 할당된 객체가 없으면 EMPTY_VALUE를 반환합니다.

Angle (Set Method)

"Angle" 속성의 값을 설정합니다.

```
bool  Angle(
   double  angle      // 속성 값
   )
```

Parameters

angle

[in] "Angle" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Font (Get Method)

"Font" 속성 값을 가져옵니다.

```
string  Font() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Font" 속성 값. 할당된 객체가 없다면, ""을 반환합니다.

Font (Set Method)

"Font" 속성의 새 값을 설정합니다.

```
bool  Font(
   string  font      // 속성 값
   )
```

Parameters

font

[in] "Font" 속성의 새 값.

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

FontSize (Get Method)

"Font size" 속성 값을 가져옵니다.

```
int  FontSize() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "FontSize" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

FontSize (Set Method)

"Font size" 속성의 새 값을 설정합니다.

```
bool  FontSize(
   int  size      // 속성 값
   )
```

Parameters

size

[in] "Font size" 속성의 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Anchor (Get Method)

"Anchor" 속성 값을 가져옵니다.

```
ENUM_ANCHOR_POINT  Anchor() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Anchor" 속성 값입니다. 할당된 개체가 없으면 WORK_VALUE 를 반환합니다.

Anchor (Set Method)

"Anchor" 속성의 새 값을 설정합니다.

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // 속성 값
   )
```

Parameters

anchor

[in] "Anchor" 속성의 새 값.

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려 있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_TEXT for CChartObjectText).

© 2000-2025, MetaQuotes Ltd.

CChartObjectLabel

CChartObjectLabel은 "레이블" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectLabel 클래스는 "레이블" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectLabel : public CChartObjectText
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel

직계 하위

CChartObjectEdit, CChartObjectRectLabel

그룹별 클래스 메서드

Create	
Create	"레이블" 그래픽 개체를 만듭니다.
Properties	
X_Distance	"X_Distance" 속성 가져오기/설정
Y_Distance	"Y_Distance" 속성 가져오기/설정
X_Size	"X_Size" 속성 가져오기/설정
Y_Size	"Y_Size" 속성 가져오기/설정
Corner	"Corner" 속성 가져오기/설정
Time	시간 좌표 변화 "Stub"
Price	가격 좌표 변화 "Stub"
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectText 클래스에서 상속된 메서드

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

더보기

객체 유형, 객체 속성, 차트 각도, 객체 바인딩 메서드, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"레이블" 그래픽 개체를 만듭니다.

```
bool  Create(
   long     chart_id,      // 차트 식별자
   string   name,          // 오브젝트명
   int      window,        // 차트 창
   int      X,             // X 좌표
   int      Y              // Y 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

X

[in] X 좌표.

Y

[in] Y 좌표.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

X_Distance (Get Method)

"X_Distance" 속성 값을 가져옵니다.

```
int  X_Distance() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "X_Distance" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

X_Distance (Set Method)

"X_Distance" 속성의 새 값을 설정합니다.

```
bool  X_Distance(
   int  X      // 속성 값
   )
```

Parameters

X

[in] "X_Distance" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (Get Method)

"Y_Distance" 속성 값을 가져옵니다.

```
int  Y_Distance() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Y_Distance" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

Y_Distance (Set Method)

"Y_Distance" 속성의 새 값을 설정합니다.

```
bool  Y_Distance(
   int  Y      // 속성 값
   )
```

Parameters

Y

[in] "Y_Distance" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

X_Size

"X_Size" 속성 값을 가져옵니다.

```
int  X_Size() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "X_Size" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Y_Size 속성 값을 가져옵니다.

```
int  Y_Size() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Y_Size" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Corner (Get Method)

"코너" 속성 값을 가져옵니다.

```
ENUM_BASE_CORNER  Corner() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "코너" 속성 값입니다. 할당된 개체가 없으면 WORK_VALUE 를 반환합니다.

Corner (Set Method)

"코너" 속성의 새 값을 설정합니다.

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // 속성 값
   )
```

Parameters

corner

[in] "코너" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Time

시간 좌표를 변경할 수 없습니다.

```
bool  Time(
   datetime  time      // 아무 값
   )
```

Parameters

time

[in] datetime 유형의 아무 값.

반환 값

always false.

© 2000-2025, MetaQuotes Ltd.

Price

가격 좌표의 변경을 금지합니다.

```
bool  Price(
   double  price      // 아무 값
   )
```

Parameters

price

[in] double 유형의 아무 값.

반환 값

always false.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_LABEL for CChartObjectLabel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectEdit

CChartObjectEdit는 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectEdit 클래스는 "편집" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectEdit : public CChartObjectLabel
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

상속 계층

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit

직계 하위

CChartObjectButton

그룹별 클래스 메서드

Create	
Create	"편집" 그래픽 개체를 생성합니다.
Properties	
TextAlign	"TextAlign" 속성 가져오기/설정
X_Size	"X Size" 속성을 가져옵니다
Y_Size	"Y Size" 속성을 가져옵니다
BackColor	"배경 색" 속성 가져오기/설정
BorderColor	"테두리 색" 속성 가져오기/설정
ReadOnly	"읽기 전용" 속성 가져오기/설정
Angle	"각도" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드

© 2000-2025, MetaQuotes Ltd.

Create	
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectText 클래스에서 상속된 메서드

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

CChartObjectLabel 클래스에서 상속된 메서드

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

더보기

객체 유형, 객체 속성, 차트 각도, 객체 바인딩 메서드, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"편집" 그래픽 개체를 만듭니다.

```
bool  Create(
   long     chart_id,      // 차트 식별자
   string   name,          // 오브젝트명
   int      window,        // 차트 창
   int      X,             // X 좌표
   int      Y,             // Y 좌표
   int      sizeX,         // X 크기
   int      sizeY          // Y 크기
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

X

[in] X 좌표.

Y

[in] Y 좌표.

sizeX

[in] X 크기.

sizeY

[in] Y 크기.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

TextAlign (Get method)

"TextAlign" 속성(텍스트 정렬 모드)값을 가져옵니다.

```
ENUM_ALIGN_MODE  TextAlign() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "TextAlign" 속성 값.

TextAlign (Set method)

"TextAlign" 속성(텍스트 정렬 모드)값을 설정합니다.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // 속성 값
   )
```

Parameters

align

[in] "TextAlign" 속성의 새 값.

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

X_Size

"X_Size" 속성의 값을 설정합니다.

```
bool  X_Size(
   int  size      // 속성 값
   )
```

Parameters

size

[in] "X_Size" 속성의 새 값.

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

Y_Size

"Y_Size" 속성의 값을 설정합니다.

```
bool  Y_Size(
   int  size      // 속성 값
   )
```

Parameters

size

[in] "Y_Size" 속성의 새 값.

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

BackColor (Get Method)

"BackColor" 속성 값을 가져옵니다.

```
color  BackColor() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "BackColor" 속성 값. 할당된 개체가 없으면 CLR_NONE을 반환합니다.

BackColor (Set Method)

"BackColor" 속성 값을 설정합니다.

```
bool  BackColor(
   color  new_color      // 속성 값
   )
```

Parameters

new_color

[in] "BackColor" 속성의 새 값.

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

BorderColor (Get Method)

"테두리 색상" 속성 값을 가져옵니다.

```
color  BorderColor() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "테두리 색상" 속성 값. 할당된 개체가 없으면 CLR_NONE을 반환합니다.

BorderColor (Set Method)

테두리 색상의 새 값을 설정합니다.

```
bool  BorderColor(
   color  new_color      // 새 속성 값
   )
```

Parameters

new_color

[in] "테두리 색상"의 새 값.

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

ReadOnly (Get Method)

"읽기 전용" 속성 값을 가져옵니다.

```
bool  ReadOnly() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "읽기 전용" 속성 값. 할당된 개체가 없으면 false를 반환합니다.

ReadOnly (Set Method)

"읽기 전용" 속성 값을 설정합니다.

```
bool  ReadOnly(
   const  bool  flag      // 새 속성 값
   )
```

Parameters

flag

[in] "읽기 전용" 속성의 새 값(true는 텍스트 편집이 비활성화되었음을 의미).

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

Angle

"각도" 속성을 변경할 수 없습니다.

```
bool  Angle(
   double  angle      // 아무 값
   )
```

Parameters

angle

[in] double 유형의 아무 값.

반환 값

always false.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_EDIT for CChartObjectEdit).

© 2000-2025, MetaQuotes Ltd.

CChartObjectButton

CChartObjectButton은 "버튼" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectButton 클래스는 "버튼" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectButton : public CChartObjectEdit
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

상속 계층

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit
CChartObjectButton

직계 하위

CChartObjectPanel

그룹별 클래스 메서드

Create	
Create	CChartObjectEdit 클래스로부터 상속됨
Properties	
State	버튼 상태 가져오기/설정 (눌림/눌림해제).
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectText 클래스에서 상속된 메서드

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

CChartObjectLabel 클래스에서 상속된 메서드

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

CChartObjectEdit 클래스에서 상속된 메서드

X_Size, Y_Size, BackColor, BackColor, BorderColor, BorderColor, ReadOnly, ReadOnly, TextAlign, TextAlign, Angle, Create

더보기

객체 유형, 객체 속성, 차트 각도, 객체 바인딩 메서드, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

State (Get Method)

"상태" 속성 값을 가져옵니다.

```
bool  State() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "상태" 속성 값. 할당된 개체가 없으면 false를 반환합니다.

State (Set Method)

"상태" 속성의 새 값을 설정합니다.

```
bool  State(
   bool  state      // 속성 값
   )
```

Parameters

X

[in] "상태" 속성의 새 값.

반환 값

true – 성공, false – 속성 변화 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_BUTTON for CChartObjectButton).

© 2000-2025, MetaQuotes Ltd.

CChartObjectSubChart

CChartObjectSubChart는 "Chart" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectSubChart 클래스는 "차트" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectSubChart : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectSubChart

그룹별 클래스 메서드

Create	
Create	"차트" 그래픽 개체를 만듭니다
Properties	
X_Distance	"X_Distance" 속성 가져오기/설정
Y_Distance	"Y_Distance" 속성 가져오기/설정
Corner	"Corner" 속성 가져오기/설정
X_Size	"X_Size" 속성 가져오기/설정
Y_Size	"Y_Size" 속성 가져오기/설정
Symbol	"심볼" 속성 가져오기/설정
Period	"기간" 속성 가져오기/설정
Scale	"스케일" 속성 가져오기/설정
DateScale	"날짜 스케일 표시" 속성 가져오기/설정
PriceScale	"가격 스케일 표시" 속성 가져오기/설정
Time	시간 좌표 변화 "Stub"
Price	가격 좌표 변화 "Stub"
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법

© 2000-2025, MetaQuotes Ltd.

Create	
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

더보기

객체 유형, 객체 속성, 차트 각도, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"SubChart" 그래픽 개체를 만듭니다.

```
bool  Create(
   long     chart_id,      // 차트 식별자
   string   name,          // 오브젝트명
   int      window,        // 차트 창
   int      X,             // X 좌표
   int      Y,             // Y 좌표
   int      sizeX,         // X 크기
   int      sizeY          // Y 크기
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

X

[in] X 좌표.

Y

[in] Y 좌표.

sizeX

[in] X 크기.

sizeY

[in] Y 크기.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

X_Distance (Get Method)

"X_Distance" 속성 값을 가져옵니다.

```
int  X_Distance() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "X_Distance" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

X_Distance (Set Method)

"X_Distance" 속성의 새 값을 설정합니다.

```
bool  X_Distance(
   int  X      // 속성 값
   )
```

Parameters

X

[in] "X_Distance" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (Get Method)

"Y_Distance" 속성 값을 가져옵니다.

```
int  Y_Distance() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Y_Distance" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

Y_Distance (Set Method)

"Y_Distance" 속성의 새 값을 설정합니다.

```
bool  Y_Distance(
   int  Y      // 속성 값
   )
```

Parameters

Y

[in] "Y_Distance" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Corner (Get Method)

"코너" 속성 값을 가져옵니다.

```
ENUM_BASE_CORNER Corner() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "코너" 속성 값입니다. 할당된 개체가 없으면 WORK_VALUE 를 반환합니다.

Corner (Set Method)

"코너" 속성의 새 값을 설정합니다.

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // 속성 값
   )
```

Parameters

corner

[in] "코너" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

X_Size (Get Method)

"X_Size" 속성 값을 가져옵니다.

```
int  X_Size() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "X_Size" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

X_Size (Set Method)

"X_Size" 속성의 새 값을 설정합니다.

```
bool  X_Size(
   int  X      // 속성 값
   )
```

Parameters

X

[in] "X_Size" 속성의 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Y_Size (Get Method)

Y_Size 속성 값을 가져옵니다.

```
int  Y_Size() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Y_Size" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

Y_Size (Set Method)

"Y_Size" 속성의 새 값을 설정합니다.

```
bool  Y_Size(
   int  Y      // 속성 값
   )
```

Parameters

Y

[in] "Y_Size" 속성의 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Symbol (Get Method)

"Symbol" 속성 값을 가져옵니다.

```
string  Symbol() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Symbol" 속성 값. 할당된 객체가 없다면, ""을 반환합니다.

Symbol (Set Method)

"Symbol" 의 새 값을 설정합니다.

```
bool  Symbol(
   string  symbol      // 심볼
   )
```

Parameters

symbol

[in] "Symbol" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Period (Get Method)

"기간" 속성 값을 가져옵니다.

```
int  Period() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "기간" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

Period (Set Method)

"기간" 속성의 새 값을 설정합니다.

```
bool  Period(
   int  period      // 기간
   )
```

Parameters

period

[in] "Period" 속성의 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Scale (Get Method)

"스케일" 속성 값을 가져옵니다.

```
double  Scale() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Scale" 속성 값. 할당된 개체가 없으면 EMPTY_VALUE를 반환합니다.

Scale (Set Method)

"스케일" 속성용 새 값을 설정합니다.

```
bool  Scale(
   double  scale      // 속성 값
   )
```

Parameters

scale

[in] "스케일" 속성용 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

DateScale (Get Method)

"DateScale" 플래그 값을 가져옵니다.

```
bool  DateScale() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "DateScale" 플래그 값. 할당된 개체가 없으면 false를 반환합니다.

DateScale (Set Method)

"DateScale" 속성의 새 값을 설정합니다.

```
bool  DateScale(
   bool  scale      // 플래그 값
   )
```

Parameters

scale

[in] "DateScale" 플래그 값.

반환 값

true – 성공, false – 플래그 변경 실패.

© 2000-2025, MetaQuotes Ltd.

PriceScale (Get Method)

"PriceScale" 플래그 값을 가져옵니다.

```
bool  PriceScale() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "PriceScale" 플래그 값. 할당된 개체가 없으면 false를 반환합니다.

PriceScale (Set Method)

"PriceScale" 플래그의 새 값을 설정합니다.

```
bool  PriceScale(
   bool  scale      // 플래그 값
   )
```

Parameters

scale

[in] "PriceScale" 플래그의 새 값.

반환 값

true - 성공, false - 플래그 변경 실패.

© 2000-2025, MetaQuotes Ltd.

Time

시간 좌표를 변경할 수 없습니다.

```
bool  Time(
   datetime  time      // 아무 값
   )
```

Parameters

time

[in]

반환 값

always false.

© 2000-2025, MetaQuotes Ltd.

Price

가격 좌표의 변경을 금지합니다.

```
bool  Price(
   double  price      // 아무 값
   )
```

Parameters

price

[in]

반환 값

always false.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_CHART for CChartObjectSubChart).

© 2000-2025, MetaQuotes Ltd.

CChartObjectBitmap

CChartObjectBitmap은 "비트맵" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectBitmap 클래스는 "비트맵" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectBitmap : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectBitmap

그룹별 클래스 메서드

Create	
Create	"비트맵" 그래픽 객체를 생성합니다
Properties	
BmpFile	"BMP Filename" 속성 가져오기/설정
X_Offset	"X_Offset" 속성 가져오기/설정
Y_Offset	"Y_Offset" 속성 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

더보기

객체 유형, 객체 속성, 그래픽 객체

© 2000–2025, MetaQuotes Ltd.

Create

"비트맵" 그래픽 객체를 생성합니다.

```
bool  Create(
   long      chart_id,      // 차트 식별자
   string    name,          // 객체명
   int       window,        // 차트 창
   datetime  time,          // 시간 좌표
   double    price          // 가격 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 – 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 – 메인 창).

time

[in] 고정점 시간 좌표.

price

[in] 고정점 가격 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

BmpFile (Get Method)

"BmpFile" 속성 값을 가져옵니다.

```
string  BmpFile() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "BmpFile" 속성 값. 할당된 개체가 없으면 false를 반환합니다.

BmpFile (Set Method)

"BmpFile" 속성의 새 값을 설정합니다.

```
bool  BmpFile(
   string  name      // 속성 값
   )
```

Parameters

name

[in] "BmpFile" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

X_Offset (Get Method)

CChartObjectBitmap 그래픽 객체의 "X_Offset" 속성 값(왼쪽 상단 모서리)을 가져옵니다.

```
int  X_Offset() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "X_Offset" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

X_Offset (Set Method)

CChartObjectBitmap 그래픽 객체의 "X_Offset" 속성(왼쪽 상단 모서리)에 대한 새 값을 설정합니다. 값은 원본 영상의 왼쪽 상단 모서리에 상대적인 픽셀 단위로 설정됩니다.

```
bool  X_Offset(
   int  X      // 속성 값
   )
```

Parameters

X

[in] "X_Offset" 속성의 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Y_Offset (Get Method)

CChartObjectBitmap 그래픽 객체의 "Y_Offset" 속성 값(왼쪽 상단 모서리)을 가져옵니다.

```
int  Y_Offset() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Y_Offset" 속성 값입니다. 할당된 객체가 없으면 0을 반환합니다.

Y_Offset (Set Method)

CChartObjectBitmap 그래픽 객체의 "Y_Offset" 속성(왼쪽 상단 모서리)에 대한 새 값을 설정합니다. 값은 원본 영상의 왼쪽 상단 모서리에 상대적인 픽셀 단위로 설정됩니다.

```
bool  Y_Offset(
   int  Y      // 속성 값
   )
```

Parameters

Y

[in] "Y_Offset" 속성 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_BITMAP for CChartObjectBitmap).

© 2000-2025, MetaQuotes Ltd.

CChartObjectBmpLabel

CChartObjectBmpLabel은 "비트맵 레이블" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectBmpLabel 클래스는 "비트맵 레이블" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectBmpLabel : public CChartObject
```

Title

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectBmpLabel

그룹별 클래스 메서드

Create	
Create	"BmpLabel" 그래픽 객체를 생성합니다
Properties	
X_Distance	"X_Distance" 속성 가져오기/설정
Y_Distance	"Y_Distance" 속성 가져오기/설정
X_Offset	"X_Offset" 속성 가져오기/설정
Y_Offset	"Y_Offset" 속성 가져오기/설정
Corner	"Corner" 속성 가져오기/설정
X_Size	"X_Size" 속성 가져오기/설정
Y_Size	"Y_Size" 속성 가져오기/설정
BmpFileOn	버튼 눌림 상태 (On) "BmpFileOn" 속성 가져오기/설정
BmpFileOff	버튼 눌림 상태 (Off) "BmpFileOff" 속성 가져오기/설정
State	"Button State" 속성 (눌림/눌림해제) 가져오기/설정
Time	시간 좌표 변화 "Stub"
Price	가격 좌표 변화 "Stub"
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법

© 2000-2025, MetaQuotes Ltd.

Create	
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

더보기

객체 유형, 객체 속성, 차트 각도, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

"BmpLabel" 그래픽 객체를 생성합니다.

```
bool  Create(
   long    chart_id,      // 차트 식별자
   string  name,          // 오브젝트명
   int     window,        // 차트 창
   int     X,             // X 좌표
   int     Y              // Y 좌표
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 – 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 – 메인 창).

X

[in] X 좌표.

Y

[in] Y 좌표.

반환 값

true – 성공, false – 오류.

© 2000-2025, MetaQuotes Ltd.

X_Distance (Get Method)

"X_Distance" 속성 값을 가져옵니다.

```
int  X_Distance() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "X_Distance" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

X_Distance (Set Method)

"X_Distance" 속성의 새 값을 설정합니다.

```
bool  X_Distance(
   int  X      // 속성 값
   )
```

Parameters

X

[in] "X_Distance" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Y_Distance (Get Method)

"Y_Distance" 속성 값을 가져옵니다.

```
int  Y_Distance() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Y_Distance" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

Y_Distance (Set Method)

"Y_Distance" 속성의 새 값을 설정합니다.

```
bool  Y_Distance(
   int  Y      // 속성 값
   )
```

Parameters

Y

[in] "Y_Distance" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

X_Offset (Get Method)

CCartObjectBmpLabel 그래픽 개체의 "X_Offset" 속성 값(왼쪽 상단 모서리)을 가져옵니다.

```
int  X_Offset() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "X_Offset" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

X_Offset (Set Method)

CChartObjectBitmap 그래픽 객체의 "X_Offset" 속성(왼쪽 상단 모서리)에 대한 새 값을 설정합니다. 값은 원본 영상의 왼쪽 상단 모서리에 상대적인 픽셀 단위로 설정됩니다.

```
bool  X_Offset(
   int  X      // 속성 값
   )
```

Parameters

X

[in] "X_Offset" 속성의 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Y_Offset (Get Method)

CCartObjectBmpLabel 그래픽 객체의 "Y_Offset"(왼쪽 상단 모서리)을 가져옵니다.

```
int  Y_Offset() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Y_Offset" 속성 값입니다. 할당된 객체가 없으면 0을 반환합니다.

Y_Offset (Set Method)

CCartObjectBmpLabel 그래픽 객체의 "Y_Offset" 속성(왼쪽 상단 모서리)에 대한 새 값을 설정합니다. 값은 원본 영상의 왼쪽 상단 모서리에 상대적인 픽셀 단위로 설정됩니다.

```
bool  Y_Offset(
   int  Y      // 속성 값
   )
```

Parameters

Y

[in] "Y_Offset" 속성 새 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Corner (Get Method)

"코너" 속성 값을 가져옵니다.

```
ENUM_BASE_CORNER Corner() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "코너" 속성 값입니다. 할당된 개체가 없으면 WORK_VALUE 를 반환합니다.

Corner (Set Method)

"코너" 속성의 새 값을 설정합니다.

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // 속성 값
   )
```

Parameters

corner

[in] "코너" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

X_Size

"X_Size" 속성 값을 가져옵니다.

```
int  X_Size() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "X_Size" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Y_Size

Y_Size 속성 값을 가져옵니다.

```
int  Y_Size() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "Y_Size" 속성 값. 할당된 객체가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

BmpFileOn (Get Method)

"BmpFileOn" 속성 값을 가져옵니다.

```
string  BmpFileOn() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "BmpFileOn" 속성 값. 할당된 객체가 없다면, ""을 반환합니다.

BmpFileOn (Set Method)

"BmpFileOn" 속성의 새 값을 설정합니다.

```
bool  BmpFileOn(
   string  name      // Flename
   )
```

Parameters

name

[in] "BmpFileOn" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

BmpFileOff (Get Method)

"BmpFileOff" 속성 값을 가져옵니다.

```
string  BmpFileOff() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "BmpFileOff" 속성 값. 할당된 객체가 없다면, ""을 반환합니다.

BmpFileOff (Set Method)

"BmpFileOff" 속성에 새 값을 설정합니다.

```
bool  BmpFileOff(
   string  name      // Flename
   )
```

Parameters

name

[in] "BmpFileOff" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

State (Get Method)

"상태" 속성 값을 가져옵니다.

```
bool  State() const
```

반환 값

클래스 인스턴스에 할당된 개체의 "상태" 속성 값. 할당된 개체가 없으면 false를 반환합니다.

State (Set Method)

"상태" 속성의 새 값을 설정합니다.

```
bool  State(
   bool  state      // 속성 값
   )
```

Parameters

state

[in] "상태" 속성의 새 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Time

시간 좌표를 변경할 수 없습니다.

```
bool  Time(
   datetime  time      // 아무 값
   )
```

Parameters

time

[in] datetime 유형의 아무 값.

반환 값

always false.

© 2000-2025, MetaQuotes Ltd.

Price

가격 좌표의 변경을 금지합니다.

```
bool  Price(
   double  price      // 아무 값
   )
```

Parameters

price

[in] double 유형의 아무 값.

반환 값

always false.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_BITMAP_LABEL for CChartObjectBmpLabel).

© 2000-2025, MetaQuotes Ltd.

CChartObjectRectLabel

CChartObjectRectLabel은 "사각 레이블" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

Description

CChartObjectRectLabel 클래스는 "사각 레이블" 개체 속성에 대한 액세스를 제공합니다.

Declaration

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

상속 계층

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectRectLabel

그룹별 클래스 메서드

Create	
Create	"RectLabel" 그래픽 객체 생성
Properties	
X_Size	수평 크기 설정
Y_Size	수직 크기 설정
BackColor	배경색 가져오기/설정
Angle	기울기 각도 변화 금지
BorderType	테두리 유형 가져오기/설정
입/출력	
virtual Save	파일에 쓰기 위한 가상 방법
virtual Load	파일에서 읽기 위한 가상 메서드
virtual Type	가상식별방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CChartObject 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Compare

ChartId, Window, Name, Name, NumPoints, Attach, SetPoint, Delete, Detach, Time, Time, Price, Price, Color, Color, Style, Style, Width, Width, Background, Background, Fill, Fill, Z_Order, Z_Order, Selected, Selected, Selectable, Selectable, Description, Description, Tooltip, Tooltip, Timeframes, Timeframes, CreateTime, LevelsCount, LevelsCount, LevelColor, LevelColor, LevelStyle, LevelStyle, LevelWidth, LevelWidth, LevelValue, LevelValue, LevelDescription, LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

CChartObjectText 클래스에서 상속된 메서드

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

CChartObjectLabel 클래스에서 상속된 메서드

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

더보기

객체 유형, 객체 속성, 그래픽 객체

© 2000-2025, MetaQuotes Ltd.

Create

테두리 유형 속성 값을 설정합니다.

```
bool  Create(
   long     chart_id,      // 차트 ID
   string   name,          // 오브젝트명
   int      window,        // 차트 창
   int      X,             // X 좌표
   int      Y,             // Y 좌표
   int      sizeX,         // 수평 크기
   int      sizeY          // 수직 크기
   )
```

Parameters

chart_id

[in] 차트 식별자 (0 - 현재 차트).

name

[in] 생성할 개체의 고유 이름.

window

[in] 차트 창 번호 (0 - 메인 창).

X

[in] X 좌표.

Y

[in] Y 좌표.

sizeX

[in] 수평 크기.

sizeY

[in] 수직 크기.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

X_Size

"X_Size" 속성의 값을 설정합니다..

```
bool  X_Size(
   int  size      // 속성 값
   )
```

Parameters

size

[in] 새 수평 크기 속성 값.

반환 값

true – 성공, false – 속성 변경 불가.

참고

"X_Size" 및 "Y_Size" 속성 값을 가져오려면 X_Size 및 Y_Size 메서드 –부모 CChartObjectLabel 클래스의–를 사용합니다.

© 2000–2025, MetaQuotes Ltd.

Y_Size

"Y_Size" 속성 값을 설정합니다.

```
bool  Y_Size(
   int  size      // 속성 값
   )
```

Parameters

size

[in] 새 수직 크기 속성 값.

반환 값

true – 성공, false – 속성 변경 불가.

참고

"X_Size" 및 "Y_Size" 속성 값을 가져오려면 X_Size 및 Y_Size 메서드 –부모 CChartObjectLabel 클래스의–를 사용합니다.

© 2000–2025, MetaQuotes Ltd.

BackColor

배경색 속성 값을 가져옵니다.

```
color  BackColor() const
```

반환 값

클래스 인스턴스에 할당된 개체의 배경색 속성 값입니다. 할당된 객체가 없으면 0을 반환합니다.

BackColor

배경색 속성 값을 설정합니다.

```
bool  BackColor(
   color  new_color      // 속성 값
   )
```

Parameters

new_color

[in] 새 배경색 속성 값.

반환 값

true – 성공, false – 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Angle

기울기 각도 속성을 변경할 수 없습니다.

```
bool  Angle(
   double  angle      // 아무 값
   )
```

Parameters

angle

[in] double 유형의 아무 값.

반환 값

항상 false.

© 2000-2025, MetaQuotes Ltd.

BorderType

테두리 유형 속성 값을 가져옵니다.

```
int  BorderType() const
```

반환 값

클래스 인스턴스에 할당된 개체의 테두리 유형 속성 값입니다. 할당된 개체가 없으면 0을 반환합니다.

BorderType

테두리 유형 속성 값을 설정합니다.

```
bool  BorderType(
   int  type      // 속성 값
   )
```

Parameters

type

[in] 새 테두리 유형 속성 값.

반환 값

true - 성공, false - 속성 변경 불가.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개 변수를 파일에 저장합니다.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개 변수를 로드합니다.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
   )
```

Parameters

file_handle

[in] FileOpen 함수에 의해 이미 열려있는 이진 파일의 핸들.

반환 값

true - 성공, false - 오류.

© 2000-2025, MetaQuotes Ltd.

Type

그래픽 개체 유형 식별자를 반환합니다.

```
virtual int  Type() const
```

반환 값

객체 유형 식별자 (OBJ_RECTANGLE_LABEL for CChartObjectRectangleLabel).

© 2000-2025, MetaQuotes Ltd.

사용자 지정 그래픽

이 섹션에서는 사용자 지정 그래픽 작업을 위한 도구를 제공합니다.

사용자 지정 차트, 도면 및 데이터 시각화를 쉽게 표시할 수 있습니다.

그래픽 개체 및 원형을 다양한 유형의 파이 차트 및 곡선을 그리기 위한 개별 클래스가 있습니다. 선 스타일 및 색상 변경, 채우기, 차트의 데이터 시리즈 작업 등 개체를 표시하는 다양한 옵션이 구현됩니다.

클래스	설명
CCanvas	사용자 지정 이미지 생성을 단순화하는 클래스
CChartCanvas	도면 차트 및 도면 요소의 클래스를 구현하기 위한 기본 클래스
CHistogramChart	히스토그램 플로팅 클래스
CLineChart	곡선 플로팅 클래스
CPieChart	파이 차트 플로팅 클래스

© 2000-2025, MetaQuotes Ltd.

CCanvas

CCanvas는 사용자 지정 이미지를 단순화하는 클래스입니다.

설명

CCanvas는 차트 개체에 바인딩되거나 바인딩되지 않은 그래픽 리소스를 만들고 그래픽 원형을 그리는 기능을 제공합니다.

선언

```
class CCanvas
```

제목

```
#include <Canvas\Canvas.mqh>
```

상속 계층

CCanvas

직계 후손

CChartCanvas, CFlameCanvas

그룹별 클래스 메서드

Create	
Attach	OBJ_BITMAP_LABEL 개체를 CCanvas 클래스의 인스턴스에 연결
Create	차트 개체에 바인딩하지 않고 그래픽 리소스를 생성
CreateBitmap	차트 개체에 바인딩된 그래픽 리소스 생성
CreateBitmapLabel	차트 개체에 바인딩된 그래픽 리소스 생성
Delete	그래픽 리소스를 삭제
속성	
ChartObjectName	바인딩된 차트의 개체의 이름을 가져오기
ResourceName	그래픽 리소스의 이름 가져오기
Width	그래픽 리소스의 너비를 가져오기
Height	그래픽 리소스의 높이를 가져오기
LineStyleSet	선 스타일 설정
화면의 개체를 업데이트	
Update	화면에 변경 사항을 표시
Resize	그래픽 리소스 크기 조정
지우기/색상 채우기	

© 2000-2025, MetaQuotes Ltd.

Create	
Erase	지우거나 지정된 색상으로 채우기
데이터 액세스	
PixelGet	지정된 좌표가 있는 점의 색상을 가져오기
PixelSet	지정된 좌표의 점 색상을 설정
원형 그리기	
LineVertical	수직선 그리기
LineHorizontal	수평선 그리기
Line	자유형 선 그리기
Polyline	Polyline 그리기
Polygon	Polygon 그리기
Rectangle	Rectangle 그리기
Circle	원 그리기
삼각형	삼각형 그리기
Ellipse	타원 그리기
Arc	호 그리기
Pie	타원 섹터 그리기
채워진 원형 그리기	
FillRectangle	채워진 Rectangle 그리기
FillCircle	채워진 원 그리기
FillTriangle	채워진 삼각형 그리기
FillPolygon	채워진 Polygon 그리기
FillEllipse	채워진 타원 그리기
채우기	영역 채우기
안티앨리어싱을 사용하여 원형 그리기	
PixelSetAA	픽셀 그리기
LineAA	선 그리기
PolylineAA	Polyline 그리기
PolygonAA	Polygon 그리기
TriangleAA	삼각형 그리기
CircleAA	원 그리기

© 2000-2025, MetaQuotes Ltd.

Create	
EllipseAA	타원 그리기
LineWu	선 그리기
PolylineWu	Polyline 그리기
PolygonWu	Polygon 그리기
TriangleWu	삼각형 그리기
CircleWu	원 그리기
EllipseWu	타원 그리기
LineThick	안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 자유형 선의 세그먼트 그리기.
LineThickVertical	안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 자유형 선의 수직 세그먼트 그리기.
LineThickHorizontal	안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 자유형 선의 수평 세그먼트 그릭기.
PolygonSmooth	두 개의 안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 폴리곤을 그리기
PolygonThick	안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 폴리곤 그리기
PolylineSmooth	두 개의 안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 폴리선 그리기
PolylineThick	안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 폴리선 그리기
텍스트	
FontSet	글꼴 매개변수 설정
FontNameSet	글꼴 이름 설정
FontSizeSet	글꼴 크기 설정
FontFlagsSet	글꼴 플래그 설정
FontAngleSet	글꼴 기울기 각도 설정
FontGet	글꼴 매개변수 가져오기
FontNameGet	글꼴 이름 가져오기
FontSizeGet	글꼴 크기 가져오기
FontFlagsGet	글꼴 플래그 가져오기
FontAngleGet	글꼴 기울기 각도 가져오기
TextOut	텍스트 표시
TextWidth	텍스트 너비 가져오기

© 2000-2025, MetaQuotes Ltd.

Create	
TextHeight	텍스트 높이 가져오기
TextSize	텍스트 크기 가져오기
투명도	
TransparentLevelSet	투명도 수준 설정
입력/출력	
LoadFromFile	BMP 파일에서 이미지 읽기

© 2000-2025, MetaQuotes Ltd.

Attach

OBJ_BITMAP_LABEL 개체에서 그래픽 리소스를 가져와 CCanvas 클래스의 인스턴스에 연결합니다.

```
bool  Attach(
   const long          chart_id,                          // 차트 식별자
   const string        objname,                           // 개체 이름
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // 색상 처리 방법
   );
```

리소스(OBJ_BITMAP_LABEL 개체에 대한)를 만들고 이를 CCanvas 클래스의 인스턴스에 연결합니다.

```
bool  Attach(
   const long          chart_id,                          // 차트 식별자
   const string        objname,                           // 개체 이름
   const int           width,                             // 이미지 너비(픽셀)
   const int           height,                            // 이미지 높이(픽셀)
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // 색상 처리 방법
   );
```

매개변수

chart_id

[out] 차트 식별자.

objname

[in] 그래픽 개체의 이름.

width

[in] 리소스 이미지 너비.

height

[in] 리소스 이미지 높이.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] 알파 채널 처리 방법. 알파 채널은 기본적으로 무시됩니다.

값 반환

개체 연결에 성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

Arc

(x1,y1) 및 (x2,y2)에 모서리가 있는 Rectangle에 내접한 타원의 호를 그립니다. 호의 경계선는 타원의 중심에서 좌표 (x3,y3) 및 (x4,y4)를 사용하여 두 점까지 연장되는 선으로 잘립니다.

```
void  Arc(
   int          x1,       // Rectangle 좌측 상단 모서리의 X 좌표
   int          y1,       // Rectangle 좌측 상단 모서리의 Y 좌표
   int          x2,       // Rectangle의 우측 하단 모서리의 X 좌표
   int          y2,       // Rectangle 우측 하단 모서리의 Y 좌표
   int          x3,       // 호의 경계선을 찾기 위한 첫 번째 점의 X 좌표
   int          y3,       // 호의 경계선을 찾기 위한 첫 번째 점의 Y 좌표
   int          x4,       // 호의 경계선을 찾기 위한 두 번째 점의 X 좌표
   int          y4,       // 호의 경계선을 찾기 위한 두 번째 점의 Y 좌표
   const uint   clr       // 색상
   );
```

매개변수

x1

[in] Rectangle을 이루는 좌측 상단 모서리의 X 좌표.

y1

[in] Rectangle을 이루는 좌측 상단 모서리의 Y 좌표.

x2

[in] Rectangle을 이루는 우측 하단 모서리의 X 좌표.

y2

[in] Rectangle을 이루는 우측 하단 모서리의 Y 좌표.

x3

[in] 호의 경계선을 얻기 위해 Rectangle 중심에서 선이 그려지는 첫 번째 점의 X 좌표.

y3

[in] 호의 경계선을 얻기 위해 Rectangle 중심에서 선이 그려지는 첫 번째 점의 Y 좌표.

x4

[in] 호의 경계선을 얻기 위해 Rectangle의 중심에서 선이 그려지는 두 번째 점의 X 좌표.

y4

[in] 호의 경계선을 얻기 위해 Rectangle의 중심에서 선이 그려지는 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식). 색상을 ARGB로 변환하려면 ColorToARGB() 기능을 사용합니다.

점 (x,y)에 중심으로, 반지름 rx 및 ry로 하는 타원의 호를 Rectangle에 내접하게 그립니다. 각 fi3 및 fi4로 형성된 레이를 사용하여 타원 중심에서 호 경계선을 자릅니다.

```
void  Arc(
   int          x,        // 타원 중심의 X 좌표
```

© 2000-2025, MetaQuotes Ltd.

```
   int          y,       // 타원 중심의 Y 좌표
   int          rx,      // X 축 타원 반지름
   int          ry,      // Y 축 타원 반지름
   int          fi3,     // 호의 첫 번째 경계를 정의하는 타원의 중심에서 나오는 레이 각도
   int          fi4,     // 호의 두 번째 경계를 정의하는 타원의 중심에서 나오는 레이 각도,
   const uint   clr      // 색상
   );
```

점 (x,y)에 중심으로 하고, 반지름 rx 및 ry로 하는 Rectangle에 내접하는 타원의 호를 그리고, 호 경계선의 좌표를 반환. 각 fi3 및 fi4로 형성된 레이를 사용하여 타원 중심에서 호 경계선을 자릅니다.

```
void  Arc(
   int          x,       // 타원 중심의 X 좌표
   int          y,       // 타원 중심의 Y 좌표
   int          rx,      // X 축 타원 반지름
   int          ry,      // Y 축 타원 반지름
   int          fi3,     // 호의 첫 번째 경계를 정의하는 타원의 중심에서 나오는 레이 각도
   int          fi4,     // 호의 두 번째 경계를 정의하는 타원의 중심에서 나오는 레이 각도,
   int&         x3,      // 첫 번째 호 경계 X 좌표
   int&         y3,      // 첫 번째 호 경계 Y 좌표
   int&         x4,      // 두 번째 호 경계 X 좌표
   int&         y4,      // 두 번째 호 경계 Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x

[in] 타원 중심의 X 좌표.

y

[in] 차원 중심의 Y 좌표.

rx

[in] X 축 타원 반지름(픽셀).

ry

[in] Y 축 타원 반지름(픽셀).

fi3

[in] 호의 첫 번째 경계를 정의하는 각도(라디안).

fi4

[in] 호의 두 번째 경계를 정의하는 각도(라디안).

x3

[out] 첫 번째 호 경계의 X 좌표를 가져오는 변수.

y3

[out] 첫 번째 호 경계의 Y 좌표를 가져오는 변수.

x4

© 2000-2025, MetaQuotes Ltd.

[out] 두 번째 호 경계의 X 좌표를 가져오는 변수.

y4

[out] 두 번째 호 경계의 Y 좌표를 가져오는 변수.

clr

[in] 색상(ARGB 형식). 색상을 ARGB로 변환하려면 ColorToARGB() 기능을 사용합니다.

호출 클래스 메서드 예시:

```
#include <Canvas\Canvas.mqh>
CCanvas 캔버스;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- 캔버스 생성
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("캔버스 생성 오류: ",GetLastError());
     }
//--- 캔버스 비우기
   canvas.Erase(clrWhite);
//--- Rectangle 그리기
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- 첫 번째 호 그리기
   canvas.Arc(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrRed));
   int x1,y1,x2,y2;
//--- 두 번째 호 그리기
   canvas.Arc(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,x1,y1,x2,y2,ColorToARGB(cl
//--- 호 좌표 인쇄
   PrintFormat("First point of arc at (%G,%G), second point of arc at (%G,%G)",x1,y1,x
   canvas.CircleAA(x1,y1,3, ColorToARGB(clrRed));
   canvas.CircleAA(x2,y2,3, ColorToARGB(clrBlue));
//--- 업데이트된 캔버스 표시
   canvas.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

Pie

(x1,y1) 및 (x2,y2)에 모서리가 있는 Rectangle에 새겨진 타원의 채워진 부분을 그리기. 섹터 경계는 타원의 중심에서 선으로 잘리고 좌표 (x3,y3) 및 (x4,y4)를 사용하여 두 점까지 연장됩니다.

```
void  Pie(
   int          x1,       // Rectangle 좌측 상단 모서리의 X 좌표
   int          y1,       // Rectangle 좌측 상단 모서리의 Y 좌표
   int          x2,       // Rectangle의 우측 하단 모서리의 X 좌표
   int          y2,       // Rectangle 우측 하단 모서리의 Y 좌표
   int          x3,       // 호의 경계선을 찾기 위한 첫 번째 점의 X 좌표
   int          y3,       // 호의 경계선을 찾기 위한 첫 번째 점의 Y 좌표
   int          x4,       // 호의 경계선을 찾기 위한 두 번째 점의 X 좌표
   int          y4,       // 호의 경계선을 찾기 위한 두 번째 점의 Y 좌표
   const uint   clr,      // 선 색상
   const uint   fill_clr // 색상 채우기
   );
```

매개변수

x1

[in] Rectangle을 이루는 좌측 상단 모서리의 X 좌표.

y1

[in] Rectangle을 이루는 좌측 상단 모서리의 Y 좌표.

x2

[in] Rectangle을 이루는 우측 하단 모서리의 X 좌표.

y2

[in] Rectangle을 이루는 우측 하단 모서리의 Y 좌표.

x3

[in] 호의 경계선을 얻기 위해 Rectangle 중심에서 선이 그려지는 첫 번째 점의 X 좌표.

y3

[in] 호의 경계선을 얻기 위해 Rectangle 중심에서 선이 그려지는 첫 번째 점의 Y 좌표.

x4

[in] 호의 경계선을 얻기 위해 Rectangle의 중심에서 선이 그려지는 두 번째 점의 X 좌표.

y4

[in] 호의 경계선을 얻기 위해 Rectangle의 중심에서 선이 그려지는 두 번째 점의 Y 좌표.

clr

[in] ARGB 형식 섹터 테두리 색상.

fill_clr

[in] 색더의 색상을 ARGB 형식으로 채우기. 색상을 ARGB로 변환하려면 ColorToARGB() 기능을 사용합니다.

© 2000-2025, MetaQuotes Ltd.

Rectangle에 내접하고 반지름 rx 및 ry이며 점 (x,y)에 중심을 둔 타원의 채워진 섹터를 그리기. 섹터 경계는 각도 fi3 및 fi4에 의해 형성된 광선에 의해 타원 중심에서 잘립니다.

```
void  Pie(
   int          x,        // 타원 중심의 X 좌표
   int          y,        // 타원 중심의 Y 좌표
   int          rx,       // X 축 타원 반지름
   int          ry,       // Y 축 타원 반지름
   int          fi3,      // 호의 첫 번째 경계를 정의하는 타원의 중심에서 나오는 레이 각도
   int          fi4,      // 호의 두 번째 경계를 정의하는 타원의 중심에서 나오는 레이 각도,
   const uint   clr,      // 선 색상
   const uint   fill_clr // 색상 채우기
   );
```

Rectangle에 내접하는 반지름 rx 및 ry이며 점 (x,y)에 중심을 둔 타원의 태워진 섹터를 그리고, 호 경계의 좌표도 반환합니다. 섹터 경계는 각도 fi3 및 fi4에 의해 형성된 광선에 의해 타원 중심에서 잘립니다.

```
void  Pie(
   int          x,        // 타원 중심의 X 좌표
   int          y,        // 타원 중심의 Y 좌표
   int          rx,       // X 축 타원 반지름
   int          ry,       // Y 축 타원 반지름
   int          fi3,      // 호의 첫 번째 경계를 정의하는 타원의 중심에서 나오는 레이 각도
   int          fi4,      // 호의 두 번째 경계를 정의하는 타원의 중심에서 나오는 레이 각도,
   int&         x3,       // 첫 번째 호 경계 X 좌표
   int&         y3,       // 첫 번째 호 경계 Y 좌표
   int&         x4,       // 두 번째 호 경계 X 좌표
   int&         y4,       // 두 번째 호 경계 Y 좌표
   const uint   clr,      // 선 색상
   const uint   fill_clr // 색상 채우기
   );
```

매개변수

x

[in] 타원 중심의 X 좌표.

y

[in] 차원 중심의 Y 좌표.

rx

[in] X 축 타원 반지름(픽셀).

ry

[in] X 축 타원 반지름(픽셀).

fi3

[in] 호의 첫 번째 경계를 정의하는 각도(라디안).

fi4

[in] 호의 두 번째 경계를 정의하는 각도(라디안).

© 2000-2025, MetaQuotes Ltd.

x3

[out] 첫 번째 호 경계의 X 좌표를 가져오는 변수.

y3

[out] 첫 번째 호 경계의 Y 좌표를 가져오는 변수.

x4

[out] 두 번째 호 경계의 X 좌표를 가져오는 변수.

y4

[out] 두 번째 호 경계의 Y 좌표를 가져오는 변수.

clr

[in] ARGB 형식 섹터 테두리 색상.

fill_clr

[in] 색더의 색상을 ARGB 형식으로 채우기. 색상을 ARGB로 변환하려면 ColorToARGB() 기능을 사용합니다.

호출 클래스 메서드 예시:

```
#include <Canvas\Canvas.mqh>
CCanvas 캔버스;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- 캔버스 생성
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("캔버스 생성 오류: ",GetLastError());
     }
//--- 캔버스 비우기
   canvas.Erase(clrWhite);
//--- Rectangle 그리기
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- 첫 번째 파이 그리기
   canvas.Pie(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrBlue),ColorToARGB(c
//--- 두 번째 파이 그리기
   canvas.Pie(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,ColorToARGB(clrGreen),Color
//--- 업데이트된 캔버스 표시
   canvas.Update();
   DebugBreak();
  }
```

© 2000-2025, MetaQuotes Ltd.

FillPolygon

채워진 Polygon 그리기.

```
void  FillPolygon(
   int&          x,       // Polygon 꼭지점의 X 좌표가 있는 배열
   int&          y,       // Polygon 꼭지점의 Y 좌표가 있는 배열
   const uint    clr      // 색상
   );
```

매개변수

x

[in] Polygon 꼭지점의 X 좌표 배열.

y

[in] Polygon 꼭지점의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

FillEllipse

지정된 좌표를 사용하여 사각형에 내접하는 채워진 타원을 그립니다.

```
void  FillPolygon(
   int          x1,      // Rectangle 좌측 상단 모서리의 X 좌표
   int          y1,      // Rectangle 좌측 상단 모서리의 Y 좌표
   int          x2,      // Rectangle의 우측 하단 모서리의 X 좌표
   int          y2,      // Rectangle 우측 하단 모서리의 Y 좌표
   const uint   clr       // 타원 색상
   );
```

매개변수

x1

[in] Rectangle을 이루는 좌측 상단 모서리의 X 좌표.

y1

[in] Rectangle을 이루는 좌측 상단 모서리의 Y 좌표.

x2

[in] Rectangle을 이루는 우측 하단 모서리의 X 좌표.

y2

[in] Rectangle을 이루는 우측 하단 모서리의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

GetDefaultColor

해당 인덱스에 의해 사전정의된 색상 반환.

```
static uint  GetDefaultColor(
   const uint  i      // 인덱스
   );
```

매개변수

i

[in] 색상을 가져오기 위한 인덱스.

값 반환

색상.

© 2000-2025, MetaQuotes Ltd.

ChartObjectName

바인딩된 차트 개체의 이름을 수신.

```
string  ChartObjectName();
```

값 반환

바인딩된 차트 개체 이름

© 2000-2025, MetaQuotes Ltd.

Circle

원 그리기

```
void  Circle(
   int          x,        // X 좌표
   int          y,        // Y 좌표
   int          r,        // 반지름
   const uint   clr       // 색상
   );
```

매개변수

x

[in] 원 중심의 X 좌표.

y

[in] 원 중심의 Y 좌표.

r

[in] 원 반지름.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

CircleAA

안티앨리어싱 알고리즘을 사용하여 원 그리기

```
void  CircleAA(
   const int      x,        // X 좌표
   const int      y,        // Y 좌표
   const double   r,        // 반지름
   const uint     clr       // 색상
   );
```

매개변수

x

[in] 원 중심의 X 좌표.

y

[in] 원 중심의 Y 좌표.

r

[in] 원 반지름.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

CircleWu

우(Wu)의 안티앨리어싱 알고리즘을 사용하여 원 그리기

```
void  CircleWu(
   const int      x,        // X 좌표
   const int      y,        // Y 좌표
   const double   r,        // 반지름
   const uint     clr       // 색상
   );
```

매개변수

x

[in] 원 중심의 X 좌표.

y

[in] 원 중심의 Y 좌표.

r

[in] 원 반지름.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

Create

차트 개체에 바인딩하지 않고 그래픽 리소스를 생성.

```
virtual bool  Create(
   const string       name,                                  // 이름
   const int          width,                                 // 너비
   const int          height,                                // 높이
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // 유형
   );
```

매개변수

name

[in] 그래픽 리소스 이름 기준. 생성 중에 의사 임의 문자열을 추가하여 리소스 이름을 생성합니다.

width

[in] 너비(X 축 방향 픽셀 크기).

height

[in] 높이(Y 축 방향 픽셀 크기).

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

값 반환

성공하면 true, 그렇지 않으면 false

© 2000-2025, MetaQuotes Ltd.

CreateBitmap

차트 개체에 바인딩된 그래픽 리소스를 생성.

1. 현재 차트의 기본 창에 그래픽 리소스를 생성.

```
bool  CreateBitmap(
   const string       name,                                  // 이름
   const datetime     time,                                  // 시간
   const double       price,                                 // 가격
   const int          width,                                 // 너비
   const int          height,                                // 높이
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // 유형
   );
```

2. 차트 ID 및 하위 창 숫자를 사용하여 그래픽 리소스를 생성.

```
bool  CreateBitmap(
   const long         chart_id,                              // 차트 ID
   const int          subwin,                                // 하위 창 숫자
   const string       name,                                  // 이름
   const datetime     time,                                  // 시간
   const double       price,                                 // 가격
   const int          width,                                 // 너비
   const int          height,                                // 높이
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // 유형
   );
```

매개변수

chart_id

[in] 개체 생성을 위한 차트 ID.

subwin

[in] 개체 생성을 위한 차트 하위 창 숫자.

name

[in] 차트 개체 이름 및 그래픽 리소스 이름이 기준.

time

[in] 차트 개체 고정점 시간 좌표.

가격

[in] 차트 개체 고정점 가격 좌표.

width

[in] 그래픽 리소스 너비(X 축 방향 픽셀 크기).

height

[in] 그래픽 리소스 너비(Y 축 방향 픽셀 크기).

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2025, MetaQuotes Ltd.

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

값 반환

성공하면 true, 그렇지 않으면 false

참고

첫 번째 기능 버전이 사용되는 경우 개체가 현재 차트의 기본 창에 생성됩니다.

개체 크기가 그래픽 리소스 크기와 일치합니다.

© 2000-2025, MetaQuotes Ltd.

CreateBitmapLabel

차트 개체에 바인딩된 그래픽 리소스를 생성.

1. 현재 차트의 기본 창에 그래픽 리소스를 생성.

```
bool  CreateBitmapLabel(
   const string       name,                                  // 이름
   const int          x,                                     // X 좌표
   const int          y,                                     // Y 좌표
   const int          width,                                 // 너비
   const int          height,                                // 높이
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // 유형
   );
```

2. 차트 ID 및 하위 창 숫자를 사용하여 그래픽 리소스를 생성.

```
bool  CreateBitmapLabel(
   const long         chart_id,                              // 차트 ID
   const int          subwin,                                // 하위 창 숫자
   const string       name,                                  // 이름
   const int          x,                                     // X 좌표
   const int          y,                                     // Y 좌표
   const int          width,                                 // 너비
   const int          height,                                // 높이
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // 유형
   );
```

매개변수

chart_id

[in] 개체 생성을 위한 차트 ID.

subwin

[in] 개체 생성을 위한 차트 하위 창 숫자.

name

[in] 차트 개체 이름 및 그래픽 리소스 이름이 기준.

x

[in] 차트 개체 고정점 X 좌표.

y

[in] 차트 개체 고정점 Y 좌표.

width

[in] 그래픽 리소스 너비(X 축 방향 픽셀 크기).

height

[in] 그래픽 리소스 너비(Y 축 방향 픽셀 크기).

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

© 2000-2025, MetaQuotes Ltd.

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

값 반환

성공하면 true, 그렇지 않으면 false

참고

첫 번째 기능 버전이 사용되는 경우 개체가 현재 차트의 기본 창에 생성됩니다.

개체 크기가 그래픽 리소스 크기와 일치합니다.

© 2000-2025, MetaQuotes Ltd.

Delete

그래픽 리소스를 삭제합니다.

```
void  Destroy();
```

참고

그래픽 리소스가 차트 개체에 바인딩된 경우 후자가 삭제됩니다.

© 2000-2025, MetaQuotes Ltd.

Ellipse

두 점을 사용하여 타원 그리기.

```
void  Ellipse(
   int          x1,      // X 좌표
   int          y1,      // Y 좌표
   int          x2,      // X 좌표
   int          y2,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] 타원을 이루는 첫 번째 점의 X 좌표.

y1

[in] 타원을 이루는 첫 번째 점의 Y 좌표.

x2

[in] 타원을 이루는 두 번째 점의 X 좌표.

y2

[in] 타원을 이루는 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

EllipseAA

안티앨리어싱 알고리즘을 사용하여 두 점을 기준으로 타원을 그립니다.

```
void  EllipseAA(
   int          x1,      // X 좌표
   int          y1,      // Y 좌표
   int          x2,      // X 좌표
   int          y2,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] 타원을 이루는 첫 번째 점의 X 좌표.

y1

[in] 타원을 이루는 첫 번째 점의 Y 좌표.

x2

[in] 타원을 이루는 두 번째 점의 X 좌표.

y2

[in] 타원을 이루는 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

EllipseWu

우(Wu)의 안티앨리어싱 알고리즘을 사용하여 두 점을 기준으로 타원을 그립니다.

```
void  EllipseWu(
   int          x1,      // X 좌표
   int          y1,      // Y 좌표
   int          x2,      // X 좌표
   int          y2,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] 타원을 이루는 첫 번째 점의 X 좌표.

y1

[in] 타원을 이루는 첫 번째 점의 Y 좌표.

x2

[in] 타원을 이루는 두 번째 점의 X 좌표.

y2

[in] 타원을 이루는 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

Erase

지정된 색상으로 채우거나 지우기.

```
void  Erase(
   const uint  clr=0      // 색상
   );
```

매개변수

clr=0

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

채우기

영역 채우기.

```
void  Fill(
   int          x,        // X 좌표
   int          y,        // Y 좌표
   const uint   clr       // 색상
   );
```

매개변수

x

[in] 채우기 시작점 X 좌표.

y

[in] 채우기 시작점 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

FillCircle

채워진 원 그리기.

```
void  FillCircle(
   int          x,        // X 좌표
   int          y,        // Y 좌표
   int          r,        // 반지름
   const uint   clr       // 색상
   );
```

매개변수

x

[in] 채워진 원 중심 X 좌표.

y

[in] 채워진 원 중심 Y 좌표.

r

[in] 채워진 원 반지름.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

FillRectangle

채워진 Rectangle 그리기.

```
void  FillRectangle(
   int          x1,      // X 좌표
   int          y1,      // Y 좌표
   int          x2,      // X 좌표
   int          y2,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] Rectangle을 이루는 첫 번째 점의 X 좌표.

y1

[in] Rectangle을 이루는 첫 번째 점의 Y 좌표.

x2

[in] Rectangle을 이루는 두 번째 점의 X 좌표.

y2

[in] Rectangle을 이루는 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

FillTriangle

채워지 삼각형을 그리기.

```
void  FillTriangle(
   int          x1,      // X 좌표
   int          y1,      // Y 좌표
   int          x2,      // X 좌표
   int          y2,      // Y 좌표
   int          x3,      // X 좌표
   int          y3,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] 삼각형의 첫 번째 점 X 좌표.

y1

[in] 삼각형의 첫 번째 점 Y 좌표.

x2

[in] 삼각형의 두 번째 점 X 좌표.

y2

[in] 삼각형의 두 번째 점 Y 좌표.

x3

[in] 삼각형의 세 번째 점 X 좌표.

y3

[in] 삼각형의 세 번째 점 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

FontAngleGet

글꼴 기울기 각도를 수신.

```
uint  FontAngleGet();
```

값 반환

글꼴 기울기 각도

© 2000-2025, MetaQuotes Ltd.

FontAngleSet

글꼴 기울기 각도 설정.

```
bool  FontAngleSet(
   uint  angle      // 각도
   );
```

매개변수

각도

[in] 글꼴 기울기 각도(10분의 1도).

값 반환

성공하면 true, 그렇지 않으면 false

© 2000-2025, MetaQuotes Ltd.

FontFlagsGet

글꼴 플래그 수신.

```
uint  FontFlagsGet();
```

값 반환

글꼴 플래그

© 2000-2025, MetaQuotes Ltd.

FontFlagsSet

Sets font flags.

```
bool  FontFlagsSet(
   uint  flags      // 플래그
   );
```

매개변수

flags

[in] 글꼴 생성 플래그. 플래그에 대한 자세한 내용은 TextSetFont() 함수 설명을 참조하십시오.

값 반환

성공하면 true, 그렇지 않으면 false

© 2000-2025, MetaQuotes Ltd.

FontGet

현재 글꼴 매개변수를 수신.

```
void  FontGet(
   string&  name,      // 이름
   int&     size,      // 크기
   uint&    flags,     // 플래그
   uint&    angle      // 기울기 각도
   );
```

매개변수

name

[out] 글꼴 이름을 반환하는 변수에 대한 참조.

Size

[out] 글꼴 크기를 반환하는 변수에 대한 참조.

flags

[out] 글꼴 플래그를 반환하는 변수에 대한 참조.

각도

[out] 글꼴 기울기 각도를 반환하는 변수에 대한 참조.

© 2000-2025, MetaQuotes Ltd.

FontNameGet

글꼴 이름 수신.

```
string  FontNameGet();
```

값 반환

글골 이름

© 2000-2025, MetaQuotes Ltd.

FontNameSet

글골 이름 설정.

```
bool  FontNameSet(
   string  name      // 이름
   );
```

매개변수

name

[in] 글꼴 이름. 예를 들어, "Arial".

값 반환

성공하면 true, 그렇지 않으면 false

© 2000-2025, MetaQuotes Ltd.

FontSet

현재 글꼴 설정.

```
bool  FontSet(
   const string  name,        // 이름
   const int     size,        // 크기
   const uint    flags=0,     // 플래그
   const uint    angle=0      // 각도
   );
```

매개변수

name

[in] 글꼴 이름. 예를 들어, "Arial".

Size

[in] 글꼴 크기. 크기 설정에 대한 자세한 내용은 TextSetFont() 함수 설명을 참조합니다.

flags=0

[in] 글꼴 생성 플래그. 플래그에 대한 자세한 내용은 TextSetFont() 함수 설명을 참조하십시오.

angle=0

[in] 글꼴 기울기 각도(10분의 1도).

값 반환

성공하면 true, 그렇지 않으면 false

© 2000-2025, MetaQuotes Ltd.

FontSizeGet

글꼴 크기 수신.

```
int  FontSizeGet();
```

값 반환

글꼴 크기

© 2000-2025, MetaQuotes Ltd.

FontSizeSet

글꼴 크기 설정.

```
bool  FontSizeSet(
   int  size      // 크기
   );
```

매개변수

Size

[in] 글꼴 크기. 크기 설정에 대한 자세한 내용은 TextSetFont() 함수 설명을 참조합니다.

값 반환

성공하면 true, 그렇지 않으면 false

© 2000-2025, MetaQuotes Ltd.

Height

그래픽 리소스의 높이를 수신.

```
int  Height();
```

값 반환

그래필 리소스의 높이

© 2000-2025, MetaQuotes Ltd.

Line

자유형 선의 세그먼트를 그리기.

```
void  Line(
   int          x1,      // X 좌표
   int          y1,      // Y 좌표
   int          x2,      // X 좌표
   int          y2,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] 세그먼트 첫 번째 점의 X 좌표.

y1

[in] 세그먼트 첫 번째 점의 Y 좌표.

x2

[in] 세그먼트 두 번째 점의 X 좌표.

y2

[in] 세그먼트 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

LineAA

안티앨리어싱 알고리즘을 사용하여 자유형 선의 세그먼트를 그리기.

```
void  LineAA(
   const int   x1,                  // X 좌표
   const int   y1,                  // Y 좌표
   const int   x2,                  // X 좌표
   const int   y2,                  // Y 좌표
   const uint  clr,                 // 색상
   const uint  style=UINT_MAX       // 선 스타일
   );
```

매개변수

x1

[in] 세그먼트 첫 번째 점의 X 좌표.

y1

[in] 세그먼트 첫 번째 점의 Y 좌표.

x2

[in] 세그먼트 두 번째 점의 X 좌표.

y2

[in] 세그먼트 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

style=UINT_MAX

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

LineWu

우(Wu)의 안티앨리어싱 알고리즘을 사용하여 자유형 선에 세그먼트 그리기.

```
void  LineWu(
   const int   x1,                  // X 좌표
   const int   y1,                  // Y 좌표
   const int   x2,                  // X 좌표
   const int   y2,                  // Y 좌표
   const uint  clr,                 // 색상
   const uint  style=UINT_MAX       // 선 스타일
   );
```

매개변수

x1

[in] 세그먼트 첫 번째 점의 X 좌표.

y1

[in] 세그먼트 첫 번째 점의 Y 좌표.

x2

[in] 세그먼트 두 번째 점의 X 좌표.

y2

[in] 세그먼트 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

style=UINT_MAX

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

LineHorizontal

수직선의 세그먼트 그리기.

```
void  LineHorizontal(
   int          x1,      // X 좌표
   int          x2,      // X 좌표
   int          y,       // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] 세그먼트 첫 번째 점의 X 좌표.

x2

[in] 세그먼트 두 번째 점의 X 좌표.

y

[in] 세그먼트의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

LineVertical

수직선의 세그먼트 그리기.

```
void  LineVertical(
   int          x,       // X 좌표
   int          y1,      // Y 좌표
   int          y2,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x

[in] 세그먼트의 X 좌표.

y1

[in] 세그먼트 첫 번째 점의 Y 좌표.

y2

[in] 세그먼트 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

LineStyleSet

선 스타일 설정.

```
void  LineStyleSet(
   const uint  style      // 스타일
   );
```

매개변수

스타일

[in] 선 스타일.

참고

입력 매개변수에는 ENUM_LINE_STYLE 열거 값이 포함될 수 있습니다. 또한 사용자 지정 선 그리기 스타일을 생성할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

LineThick

안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 자유형 선의 세그먼트 그리기.

```
void  LineThick(
   const int       x1,            // 세그먼트 첫 번째 점의 X 좌표
   const int       y1,            // 세그먼트 첫 번째 점의 Y 좌표
   const int       x2,            // 세그먼트 두 번째 점의 X 좌표
   const int       y2,            // 세그먼트 두번째 점의 Y 좌표
   const uint      clr,           // 색상
   const int       size,          // 선 너비
   const uint      style,         // 선 스타일
   ENUM_LINE_END   end_style      // 선 끝 스타일
  \)
```

매개변수

x1

[in] 세그먼트 첫 번째 점의 X 좌표.

y1

[in] 세그먼트 첫 번째 점의 Y 좌표.

x2

[in] 세그먼트 두 번째 점의 X 좌표.

y2

[in] 세그먼트 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

Size

[in] 선 너비.

스타일

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

end_style

[in] 선 스타일은 ENUM_LINE_END 열거값 중 하나입니다

ENUM_LINE_END

ID	설명
LINE_END_ROUND	선 끝이 둥글게 되었습니다.
LINE_END_BUTT	선 끝이 잘립니다.
LINE_END_SQUARE	선이 채워진 Rectangle으로 끝납니다.

© 2000-2025, MetaQuotes Ltd.

LineThickVertical

안티앨리어싱 알고리즘을 사용하여 지정된 폭을 가진 자유형 선의 수직 세그먼트 그리기.

```
void  LineThickVertical(
   const int       x,              // 세그먼트의 X 좌표
   const int       y1,             // 세그먼트 첫 번째 점의 Y 좌표
   const int       y2,             // 세그먼트 두번째 점의 Y 좌표
   const uint      clr,            // 색상
   const int       size,           // 선 너비
   const uint      style,          // 선 스타일
   ENUM_LINE_END   end_style       // 선 끝 스타일
  \)
```

매개변수

x
  [in] 세그먼트의 X 좌표.

y1
  [in] 세그먼트 첫 번째 점의 Y 좌표.

y2
  [in] 세그먼트 두 번째 점의 Y 좌표.

clr
  [in] 색상(ARGB 형식).

Size
  [in] 선 너비.

스타일
  [in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

end_style
  [in] 선 스타일은 ENUM_LINE_END 열거 값중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

LineThickHorizontal

지정된 너비 안티엘리어싱이 있는 자유형 선의 수평 세그먼트를 그립니다.

```
void  LineThickHorizontal(
   const int       x1,            // 세그먼트 첫 번째 점의 X 좌표
   const int       x2,            // 세그먼트 두 번째 점의 X 좌표
   const int       y,             // 세그먼트의 Y 좌표
   const uint      clr,           // 색상
   const int       size,          // 선 너비
   const uint      style,         // 선 스타일
   ENUM_LINE_END   end_style      // 선 끝 스타일
  \)
```

매개변수

x1

[in] 세그먼트 첫 번째 점의 X 좌표.

x2

[in] 세그먼트 두 번째 점의 X 좌표.

y

[in] 세그먼트의 Y 좌표.

clr

[in] 색상(ARGB 형식).

Size

[in] 선 너비.

스타일

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

end_style

[in] 선 스타일은 ENUM_LINE_END 열거 값중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

LoadFromFile

BMP 파일에서 이미지를 읽습니다.

```
bool  LoadFromFile(
   const string  filename      // 파일 이름
   );
```

매개변수

Flename

[in] 파일 이름("BMP" 확장자 포함).

값 반환

성공하면 true, 그렇지 않으면 false

© 2000-2025, MetaQuotes Ltd.

PixelGet

지정된 좌표의 점 색상 수신.

```
uint  PixelGet(
   const int  x,     // X 좌표
   const int  y      // Y 좌표
   );
```

매개변수

x

[in] 점의 X 좌표.

y

[in] 점의 Y 좌표.

값 반환

점 색상 (ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

PixelSet

지정된 좌표의 점 색상을 설정.

```
void  PixelSet(
   const int   x,        // X 좌표
   const int   y,        // Y 좌표
   const uint  clr       // 색상
   );
```

매개변수

x

[in] 점의 X 좌표.

y

[in] 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

PixelSetAA

안티앨리어싱 알고리즘을 이용하여 점 그리기.

```
void  PixelSetAA(
   const double  x,        // X 좌표
   const double  y,        // Y 좌표
   const uint    clr       // 색상
   );
```

매개변수

x

[in] 점의 X 좌표.

y

[in] 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

Polygon

Polygon 그리기.

```
void  Polygon(
   int&         x[],     // X 좌표의 배열
   int&         y[],     // Y 좌표의 배열
   const uint   clr      // 색상
   );
```

매개변수

x[]

[in] Polygon 꼭지점들의 X 좌표 배열.

y[]

[in] Polygon 꼭지점들의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

PolygonAA

안티앨리어싱 알고리즘을 이용하여 Polygon 그리기.

```
void  PolygonAA(
   int&         x[],                 // X 좌표 배열
   int&         y[],                 // Y 좌표 배열
   const uint   clr,                 // 색상
   const uint   style=UINT_MAX       // 선 스타일
   );
```

매개변수

x[]

[in] Polygon 꼭지점들의 X 좌표 배열.

y[]

[in] Polygon 꼭지점들의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

style=UINT_MAX

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

PolygonWu

우(Wu)의 안티앨리어싱 알고리즘을 사용하여 Polygon 그리기.

```
void  PolygonWu(
   int&         x[],                // X 좌표 배열
   int&         y[],                // Y 좌표 배열
   const uint   clr,                // 색상
   const uint   style=UINT_MAX      // 선 스타일
   );
```

매개변수

x[]

[in] Polygon 꼭지점들의 X 좌표 배열.

y[]

[in] Polygon 꼭지점들의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

style=UINT_MAX

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

PolygonThick

안티앨리어싱 알고리즘을 사용하여 지정한 너비의 Polygon을 그리기.

```
void  PolygonThick(
   const int&      x[],          // Polygon 꼭지점의 X 좌표 배열
   const int&      y[],          // Polygon 꼭지점의 Y 좌표 배열
   const uint      clr,          // 색상
   const int       size,         // 선 너비
   const uint      style,        // 선 스타일
   ENUM_LINE_END   end_style     // 선 끝 스타일
  \)
```

매개변수

x[]

[in] Polygon 꼭지점 X 좌표 배열.

y[]

[in] Polygon 꼭지점 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

Size

[in] 선 너비.

스타일

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

end_style

[in] 선 스타일은 ENUM_LINE_END 열거 값중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

PolygonSmooth

두 개의 안티앨리어싱 알고리즘을 사용하여 지정된 너비의 Polygon을 연속적으로 그립니다. 먼저, 개별 세그먼트는 베지어 곡선을 기준으로 평활됩니다. 그런 다음 래스터 안티앨리어싱 알고리즘이 렌터링 품질을 향상시키기 위해 이러한 세그먼트에서 작성된 Polygon에 적용됩니다.

```
void  PolygonSmooth(
   int&             x[],                    // Polygon 꼭지점 X 좌표 배열
   int&             y[],                    // Polygon 꼭지점 Y 좌표 배열
   const uint       clr,                    // 색상
   const int        size,                   // 선 너비
   ENUM_LINE_STYLE  style=STYLE_SOLID,      // 선 스타일
   ENUM_LINE_END    end_style=LINE_END_ROUND,  // 선 끝 스타일
   double           tension=0.5,            // 안티앨리어싱 매개변수 값
   double           step=10                 // 근사선 길이
  \)
```

매개변수

&x[]

[in] Polygon 꼭지점 X 좌표 배열.

&y[]

[in] Polygon 꼭지점 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

Size

[in] 선 너비.

style=STYLE_SOLID

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

end_style=LINE_END_ROUND

[in] 라인 스타일은 ENUM_LINE_END 열거 값 중 하나입니다.

tension=0.5

[in] 평활 매개변수 값.

step=10

[in] 근사선의 길이.

© 2000-2025, MetaQuotes Ltd.

Polyline

Polyline 그리기.

```
void  Polyline(
   int&         x[],     // X 좌표의 배열
   int&         y[],     // Y 좌표의 배열
   const uint   clr      // 색상
   );
```

매개변수

x[]

[in] Polyline의 X 좌표 배열.

y[]

[in] Polyline의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

PolylineSmooth

두 개의 안티앨리어싱 알고리즘을 사용하여 지정한 너비의 폴리선을 연속적으로 그리기. 첫째, 개별 선 세그먼트는 베지어 곡선을 기준으로 평활됩니다. 그런 다음 래스터 안티앨리어싱 알고리즘을 이러한 세그먼트에서 작성된 Polyline에 적용하여 렌더링 품질을 개선합니다.

```
void  PolylineSmooth(
   const int&        x[],                          // Polyline 점들의 X 좌표 배열
   const int&        y[],                          // Polyline 점들의 Y 좌표 배열
   const uint        clr,                          // 색상
   const int         size,                         // 선 너비
   ENUM_LINE_STYLE   style=STYLE_SOLID,            // 선 스타일
   ENUM_LINE_END     end_style=LINE_END_ROUND,     // 선 끝 스타일
   double            tension=0.5,                  // 안티앨리어싱 매개변수 값
   double            step=10                       // 근사 단계
  \)
```

매개변수

&x[]

[in] Polyline의 X 좌표 배열.

&y[]

[in] Polyline의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

Size

[in] 선 너비.

style=STYLE_SOLID

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

end_style=LINE_END_ROUND

[in] 선 스타일은 ENUM_LINE_END 열거 값중 하나입니다.

tension=0.5

[in] 평활 매개변수 값.

step=10

[in] 근사 단계.

© 2000-2025, MetaQuotes Ltd.

PolylineThick

안티앨리어싱 알고리즘을 사용하여 지정된 너비로 Polyline 그리기.

```
void  PolylineThick(
   const int       &x[],          // Polyline 점의 X 좌표 배열
   const int       &y[],          // Polyline 점의 Y 좌표 배열
   const uint      clr,           // 색상
   const int       size,          // 선 너비
   const uint      style,         // 선 스타일
   ENUM_LINE_END   end_style      // 선 끝 스타일
  \)
```

매개변수

&x[]

[in] Polyline의 X 좌표 배열.

&y[]

[in] Polyline의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

Size

[in] 선 너비.

스타일

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

end_style

[in] 선 스타일은 ENUM_LINE_END 열거 값 중 하나입니다

© 2000-2025, MetaQuotes Ltd.

PolylineWu

우(Wu)의 안티앨리어싱 알고리즘을 사용하여 폴리선을 그리기.

```
void  PolylineWu(
   int&         x[],                 // X 좌표 배열
   int&         y[],                 // Y 좌표 배열
   const uint   clr,                 // 색상
   const uint   style=UINT_MAX       // 선 스타일
   );
```

매개변수

x[]

[in] Polyline의 X 좌표 배열.

y[]

[in] Polyline의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

style=UINT_MAX

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

PolylineAA

안티앨리어싱 알고리즘을 사용하여 Polyline 그리기.

```
void  PolylineAA(
   int&         x[],                 // X 좌표 배열
   int&         y[],                 // Y 좌표 배열
   const uint   clr,                 // 색상
   const uint   style=UINT_MAX       // 선 스타일
   );
```

매개변수

x[]

[in] Polyline의 X 좌표 배열.

y[]

[in] Polyline의 Y 좌표 배열.

clr

[in] 색상(ARGB 형식).

style=UINT_MAX

[in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

Rectangle

두 점을 사용하여 Rectangle 그리기.

```
void  Rectangle(
   int          x1,      // X 좌표
   int          y1,      // Y 좌표
   int          x2,      // X 좌표
   int          y2,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] Rectangle을 이루는 첫 번째 점의 X 좌표.

y1

[in] Rectangle을 이루는 첫 번째 점의 Y 좌표.

x2

[in] Rectangle을 이루는 두 번째 점의 X 좌표.

y2

[in] Rectangle을 이루는 두 번째 점의 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

Resize

Resizes a graphical resource.

```
bool  Resize(
   const int  width,      // 너비
   const int  height      // 높이
   );
```

매개변수

width

[in] 그래픽 리소스의 새 너비.

height

[in] 그래픽 리소스의 새 높이.

값 반환

성공하면 true, 그렇지 않으면 false

참고

크기를 조정할 때 이전 이미지는 저장되지 않습니다.

© 2000-2025, MetaQuotes Ltd.

ResourceName

그래픽 리소스의 이름 수신

```
string  ResourceName();
```

값 반환

그래픽 리소스의 이름

© 2000-2025, MetaQuotes Ltd.

TextHeight

텍스트 높이 수신.

```
int  TextHeight(
   const string  text      // 텍스트
   );
```

매개변수

텍스트

[in] 측정할 텍스트.

값 반환

텍스트 높이(픽셀)

참고

현재 글골은 텍스트 측정에 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

TextOut

텍스트 표시.

```
void  TextOut(
   int          x,              // X 좌표
   int          y,              // Y 좌표
   string       text,           // 텍스트
   const uint   clr,            // 색상
   uint         alignment=0     // 정렬
   );
```

매개변수

x

[in] 텍스트 고정점의 X 좌표.

y

[in] 텍스트 고정점의 Y 좌표.

텍스트

[in] 표시될 텍스트.

clr

[in] 색상(ARGB 형식).

alignment=0

[in] 텍스트 고정 방법. 고정 방법에 대한 자세한 내용은 TextOut() 함수 설명을 참조하십시오.

참고

현재 글꼴은 텍스트를 표시하는 데 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

TextSize

텍스트 크기 수신.

```
void  TextSize(
   const string  text,        // 텍스트
   int&          width,       // 너비
   int&          height       // 높이
   );
```

매개변수

텍스트

[in] 측정할 텍스트.

width

[out] 텍스트 너비를 반환하기 위한 변수에 대한 참조입니다.

height

[out] 텍스트 높이를 반환하기 위한 변수에 대한 참조.

참고

현재 글꼴은 텍스트를 측정하는 데 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

TextWidth

텍스트 너비 수신.

```
int  TextWidth(
   const string  text      // 텍스트
   );
```

매개변수

텍스트

[in] 측정할 텍스트.

값 반환

텍스트 높이(픽셀)

참고

현재 글꼴은 텍스트를 측정하는 데 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

TransparentLevelSet

투명도 수준 설정.

```
void  TransparentLevelSet(
   const uchar  value      // 값
   );
```

매개변수

값

[in] 투명도 수준 새 값.

참고

0은 완전 투명, 255는 완전 불투명을 나타냅니다.

투명도 수준을 설정하면 이전에 그린 모든 작업에 영향을 미칩니다. 지정한 투명도 수준은 추가 구성에 영향을 미치지 않습니다.

© 2000-2025, MetaQuotes Ltd.

삼각형

삼각형 그리기.

```
void  Triangle(
   int          x1,      // X 좌표
   int          y1,      // Y 좌표
   int          x2,      // X 좌표
   int          y2,      // Y 좌표
   int          x3,      // X 좌표
   int          y3,      // Y 좌표
   const uint   clr      // 색상
   );
```

매개변수

x1

[in] 삼각형의 첫 번째 점 X 좌표.

y1

[in] 삼각형의 첫 번째 점 Y 좌표.

x2

[in] 삼각형의 두 번째 점 X 좌표.

y2

[in] 삼각형의 두 번째 점 Y 좌표.

x3

[in] 삼각형의 세 번째 점 X 좌표.

y3

[in] 삼각형의 세 번째 점 Y 좌표.

clr

[in] 색상(ARGB 형식).

© 2000-2025, MetaQuotes Ltd.

TriangleAA

안티앨리어싱 알고리즘을 사용하여 삼각형 그리기.

```
void  TriangleAA(
   const int   x1,                  // X 좌표
   const int   y1,                  // Y 좌표
   const int   x2,                  // X 좌표
   const int   y2,                  // Y 좌표
   const int   x3,                  // X 좌표
   const int   y3,                  // Y 좌표
   const uint  clr,                 // 색상
   const uint  style=UINT_MAX       // 선 스타일
   );
```

매개변수

x1
  [in] 삼각형의 첫 번째 점 X 좌표.

y1
  [in] 삼각형의 첫 번째 점 Y 좌표.

x2
  [in] 삼각형의 두 번째 점 X 좌표.

y2
  [in] 삼각형의 두 번째 점 Y 좌표.

x3
  [in] 삼각형의 세 번째 점 X 좌표.

y3
  [in] 삼각형의 세 번째 점 Y 좌표.

clr
  [in] 색상(ARGB 형식).

style=UINT_MAX
  [in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

TriangleWu

우(Wu)의 안티앨리어싱 알고리즘을 사용하여 삼각형을 그리기.

```
void  TriangleWu(
   const int   x1,                  // X 좌표
   const int   y1,                  // Y 좌표
   const int   x2,                  // X 좌표
   const int   y2,                  // Y 좌표
   const int   x3,                  // X 좌표
   const int   y3,                  // Y 좌표
   const uint  clr,                 // 색상
   const uint  style=UINT_MAX       // 선 스타일
   );
```

매개변수

x1
  [in] 삼각형의 첫 번째 점 X 좌표.

y1
  [in] 삼각형의 첫 번째 점 Y 좌표.

x2
  [in] 삼각형의 두 번째 점 X 좌표.

y2
  [in] 삼각형의 두 번째 점 Y 좌표.

x3
  [in] 삼각형의 세 번째 점 X 좌표.

y3
  [in] 삼각형의 세 번째 점 Y 좌표.

clr
  [in] 색상(ARGB 형식).

style=UINT_MAX
  [in] 선 스타일은 ENUM_LINE_STYLE 열거 값 또는 사용자 지정 값 중 하나입니다.

© 2000-2025, MetaQuotes Ltd.

Update

변경사항을 화면에 표시.

```
void  Update(
   const bool  redraw=true      // 플래그
   );
```

매개변수

redraw=true

필요 사항을 다시 그리는 차트의 플래그.

© 2000-2025, MetaQuotes Ltd.

Width

그래픽 리소스의 너비를 수신.

```
int  Width();
```

값 반환

그래픽 리소스 너비

© 2000-2025, MetaQuotes Ltd.

CChartCanvas

차트 및 해당 요소를 그리는 데 사용되는 클래스를 구현하기 위한 기본 클래스입니다.

설명

이 클래스에는 좌표축과 좌표축 표시, 차트 범례, 그리드, 배경 등의 차트의 기본 요소로 작업하는 방법이 포함됩니다. 여기서 가시성, 텍스트 색상 등의 요소 표시 옵션을 사용자 정의할 수 있습니다.

선언

```
class CChartCanvas : public CCanvas
```

제목

```
#include <Canvas\Charts\ChartCanvas.mqh>
```

상속 계층

CCanvas

CChartCanvas

직계 후손

CHistogramChart, CLineChart, CPieChart

클래스 메서드

메서드	액션
ColorBackground	배경색을 반환하고 설정.
ColorBorder	테두리 색상을 반환하고 설정.
ColorText	텍스트 색상을 반환하고 설정.
ColorGrid	그리드 색상을 반환하고 설정.
MaxData	허용되는 최대 데이터(시리즈) 양을 반환하고 설정.
MaxDescrLen	설명자의 최대 길이를 반환하고 설정.
ShowFlags	차트 요소의 가시성 플래그를 반환하고 설정.
IsShowLegend	차트에서 범례의 가시성 플래그를 반환하고 설정.
IsShowScaleLeft	좌측에 있는 값 척도의 가시성 플래그를 반환.
IsShowScaleRight	우측에 있는 값 척도의 가시성 플래그를 반환.
IsShowScaleTop	맨 위에 있는 값 척도의 가시성 플래그를 반환.
IsShowScaleBottom	가장 아래에 있는 값 스케일의 가시성 플래그를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

メソッド	アクション
IsShowGrid	차트에 있는 그리드의 가시성 플래그를 반환.
IsShowDescriptors	차트에 있는 설명자의 가시성 플래그를 반환.
IsShowPercent	차트에 있는 백분율의 가시성 플래그를 반환.
VScaleMin	값의 수직 스케일에 대한 최소값을 반환하고 설정.
VScaleMax	값의 수직 스케일에서 최대값을 반환하고 설정.
NumGrid	차트 그리드를 표시할 때 수직 스케일 눈금 수를 반환하고 설정.
DataOffset	데이터 오프셋 값을 반환하고 설정.
DataTotal	차트에 있는 총 데이터 시리즈 수를 반환.
DrawDescriptors	설명자를 그리기 위한 가상 메서드.
DrawData	지정된 인덱스에서 데이터 시리즈를 그리는 가상 메서드.
Create	그래픽 리소스를 생성하는 가상 메서드.
AllowedShowFlags	차트 요소에 허용된 가시성 플래그 집합을 설정합니다.
ShowLegend	범례의 가시성 플래그를 설정.
ShowScaleLeft	좌측 스케일의 가시성 플래그를 설정.
ShowScaleRight	우측 스케일의 가시성 플래그를 설정.
ShowScaleTop	위쪽 스케일의 가시성 플래그를 설정.
ShowScaleBottom	아래쪽 축척에 대한 가시성 플래그를 설정.
ShowGrid	그리드의 가시성 플래그를 설정.
ShowDescriptors	설명자의 가시성 플래그를 설정.
ShowValue	값에 대한 가시성 플래그를 설정.
ShowPercent	백분율에 대한 가시성 플래그를 설정.
LegendAlignment	범례의 텍스트 정렬을 설정.
Accumulative	시리즈에 대한 값 누적 플래그를 설정.
VScaleParams	값의 수직 스케일에 대한 매개변수를 설정.
DescriptorUpdate	지정된 포지션에 있는 시리즈 설명자 값 없데이트.
ColorUpdate	지정된 포지션에서 시리즈 색상을 업데이트.

© 2000-2025, MetaQuotes Ltd.

메서드	액션
ValuesCheck	차트를 플로팅하기 위한 내부 계산을 수행.
Redraw	차트를 다시 그리기.
DrawBackground	배경 그리기.
DrawLegend	범례 다시 그리기.
DrawLegendVertical	수직 범례 그리기.
DrawLegendHorizontal	수평 범례 그리기.
CalcScales	스케일의 좌표를 계산.
DrawScales	값의 모든 스케일을 다시 그리기.
DrawScaleLeft	값의 좌측 스케일 다시 그리기.
DrawScaleRight	값의 우측 스케일 다시 그리기.
DrawScaleTop	값의 위쪽 스케일 다시 그리기
DrawScaleBottom	아래쪽 값 스케일 다시 그리기.
DrawGrid	차트를 다시 그리기.
DrawChart	차트를 다시 그리기.

클래스 CCanvas에서 상속된 메서드

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

© 2000–2025, MetaQuotes Ltd.

ColorBackground (Get method)

배경 색상 반환.

```
uint  ColorBackground()
```

값 반환

배경 색상.

ColorBackground (Set method)

배경 색상 설정.

```
void  ColorBackground(
  const uint  value,  // 배경 색상
 \)
```

매개변수

값

[in] 배경 색상.

© 2000-2025, MetaQuotes Ltd.

ColorBorder (Get method)

테두리 섹상 반환.

```
uint  ColorBorder()
```

값 반환

테두리 색상.

ColorBorder (Set method)

테두리 색상 변경.

```
void  ColorBorder(
  const uint  value,  // 테두리 색상
 \)
```

매개변수

값

[in] 테두리 색상.

© 2000-2025, MetaQuotes Ltd.

ColorText (Get method)

텍스트 색상 반환.

```
uint  ColorText()
```

값 반환

텍스트 색상.

ColorText (Set method)

텍스트 섹상 설정.

```
void  ColorText(
  const uint  value,  // 텍스트 색상
 \)
```

매개변수

값

[in] Text color.

© 2000-2025, MetaQuotes Ltd.

ColorGrid (Get method)

그리드 색상 반환.

```
uint  ColorGrid()
```

값 반환

그리드 색상.

ColorGrid (Set method)

그리드 색상 설정.

```
void  ColorGrid(
  const uint  value,  // 그리드 색상
 \)
```

매개변수

값

[in] 그리드 색상.

© 2000-2025, MetaQuotes Ltd.

MaxData (Get method)

허용되는 최대 데이터(시리즈) 양을 반환.

```
uint  MaxData()
```

값 반환

최대 데이터 양(시리즈).

MaxData (Set method)

허용되는 최대 데이터(시리즈) 양 설정.

```
void  MaxData(
  const uint  value,  // 데이터의 양
 \)
```

매개변수

값

[in] 최대 데이터 양(시리즈).

© 2000-2025, MetaQuotes Ltd.

MaxDescrLen (Get method)

설명자의 최대 길이를 반환.

```
uint  MaxDescrLen()
```

값 반환

설명자의 최대 길이 값.

MaxDescrLen (Set method)

설명자의 최대 길이를 설정.

```
void  MaxDescrLen(
  const uint  value,  // 최대 길이
 \)
```

매개변수

값

[in] 설명자의 최대 길이 값.

© 2000-2025, MetaQuotes Ltd.

ShowFlags (Get method)

차트 요소의 표시성 플래그 반환.

```
bool  ShowFlags()
```

값 반환

차트 요소의 가시성 플래그 값.

ShowFlags (Set method)

차트 요소의 가시성 플래그를 설정.

```
void  ShowFlags(
  const uint  flags,  // 플래그
 \)
```

매개변수

flags

[in] 차트 요소의 가시성 플래그 값.

© 2000-2025, MetaQuotes Ltd.

IsShowLegend

차트에서 범례의 가시성 플래그를 반환하고 설정.

```
bool  IsShowLegend()
```

값 반환

범례가 보이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleLeft

좌측에 있는 값 척도의 가시성 플래그를 반환.

```
bool  IsShowScaleLeft()
```

값 반환

값이 보이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleRight

우측에 있는 값 척도의 가시성 플래그를 반환.

```
bool  IsShowScaleRight()
```

값 반환

값이 보이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleTop

맨 위에 있는 값 척도의 가시성 플래그를 반환.

```
bool  IsShowScaleTop()
```

값 반환

값이 보이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

IsShowScaleBottom

가장 아래에 있는 값 스케일의 가시성 플래그를 반환합니다.

```
bool  IsShowScaleBottom()
```

값 반환

값이 보이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

IsShowGrid

차트에 있는 그리드의 가시성 플래그를 반환.

```
bool  IsShowGrid()
```

값 반환

그리드가 보이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

IsShowDescriptors

차트에 있는 설명자의 가시성 플래그를 반환.

```
bool  IsShowDescriptors()
```

값 반환

설명자가 보이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

IsShowPercent

차트에 있는 백분율의 가시성 플래그를 반환.

```
bool  IsShowPercent()
```

값 반환

백분율이 보이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

VScaleMin (Get method)

값의 수직 스케일에 대한 최소값을 반환.

```
double  VScaleMin()
```

값 반환

수직 스케일의 최소값.

VScaleMin (Set method)

값의 수직 스케일에 최소값을 설정.

```
void  VScaleMin(
  const double  value,    // 수직 스케일의 값
 \)
```

매개변수

값

[in] 최대값.

© 2000-2025, MetaQuotes Ltd.

VScaleMax

값의 수직 스케일에서 최대값을 반환.

```
double  VScaleMax()
```

값 반환

수직 스케일의 최대값.

VScaleMax

값의 수직 눈금에서 최대값을 설정.

```
void  VScaleMax(
  const double  value,  //   수직 스케일의 값
 \)
```

매개변수

값

[in] 최대값.

© 2000-2025, MetaQuotes Ltd.

NumGrid

차트 그리드를 플로팅할 때 수직 스케일 눈금 수를 반환.

```
uint  NumGrid()
```

값 반환

눈금 수.

NumGrid

차트 그리드를 표시할 때 수직 눈금 수를 설정.

```
void  NumGrid(
  const uint  value,  // 눈금 수
 \)
```

매개변수

값

[in] 눈금 수.

© 2000-2025, MetaQuotes Ltd.

DataOffset

데이터 오프셋 값을 반환.

```
int  DataOffset()
```

값 반환

데이터 오프셋.

DataOffset

데이터 오프셋 값 설정.

```
void  DataOffset(
  const int  value,  // 오프셋
 \)
```

매개변수

값

[in] 데이터 오프셋.

© 2000-2025, MetaQuotes Ltd.

DataTotal

차트에 있는 총 데이터 시리즈 수를 반환.

```
uint  DataTotal()
```

값 반환

시리즈 수.

© 2000-2025, MetaQuotes Ltd.

DrawDescriptors

설명자를 그리기 위한 가상 메서드.

```
virtual void  DrawDescriptors()
```

© 2000-2025, MetaQuotes Ltd.

DrawData

지정된 인덱스에서 데이터 시리즈를 그리는 가상 메서드.

```
virtual void  DrawData(
   const uint  idx=0,  // 인덱스
  \)
```

매개변수

idx=0

[in] 시리즈의 인덱스.

© 2000-2025, MetaQuotes Ltd.

Create

그래픽 리소스를 생성하는 가상 메서드.

```
virtual bool  Create(
  const string       name,      // 리소스 이름
  const int          width,     // 너비
  const int          height,    // 높이
  ENUM_COLOR_FORMAT  clrfmt,    // 형식
 \)
```

매개변수

name

[in] 그래픽 리소스 이름의 기본. 생성 중에 의사 임의 문자열을 추가하여 리소스 이름을 생성합니다.

width

[in] 너비(X 축 방향 픽셀 크기).

height

[in] 높이(Y 축 방향 픽셀 크기).

clrfmt

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

AllowedShowFlags

차트 요소에 허용된 가시성 플래그 집합을 설정합니다.

```
void  AllowedShowFlags(
  const uint  flags,  // flags
 \)
```

매개변수

flags

[in] 허용된 플래그.

© 2000-2025, MetaQuotes Ltd.

ShowLegend

범례의 가시성 플래그 값을 설정 (FLAG_SHOW_LEGEND).

```
void  ShowLegend(
  const bool  flag,  // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 범례가 표시됩니다.
- false —범례가 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

ShowScaleLeft

좌측 스케일의 가시성 플래그 값을 설정 (FLAG_SHOW_SCALE_LEFT).

```
void  ShowScaleLeft(
  const bool  flag,  // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 좌측 스케일이 표시됩니다.
- false — 좌측 스케일이 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

ShowScaleRight

우측 스케일의 가시성 플래그 값을 설정 (FLAG_SHOW_SCALE_RIGHT).

```
void  ShowScaleRight(
  const bool  flag,  // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 우측 스케일이 표시됩니다.
- false —우측 스케일이 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

ShowScaleTop

최상위 척도의 가시성 플래그 값을 설정 (FLAG_SHOW_SCALE_TOP).

```
void  ShowScaleTop(
  const bool  flag,  // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 위쪽 스케일이 표시됩니다.
- false —위쪽 스케일이 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

ShowScaleBottom

아래쪽 스케일에 대한 가시성 플래그 값을 설정 (FLAG_SHOW_SCALE_BOTTOM).

```
void  ShowScaleBottom(
  const bool  flag,     // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 아래쪽 스케일이 표시됩니다.
- false — 아래쪽 스케일이 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

ShowGrid

그리드에 대한 가시성 플래그 값을 설정(FLAG_SHOW_GRID).

```
void  ShowGrid(
  const bool  flag,  // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 그리드가 표시됩니다.
- false — 그리드가 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

ShowDescriptors

설명자의 가시성 플래그 값을 설정 (FLAG_SHOW_DESCRIPTORS).

```
void  ShowDescriptors(
  const bool  flag,    // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 설명자가 표시됩니다.
- false — 설명자가 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

ShowValue

값의 가시성 플래그를 설정 (FLAG_SHOW_VALUE).

```
void  ShowValue(
  const bool  flag,  // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 값이 표시됩니다.
- false — 값이 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

ShowPercent

백분율에 대한 가시성 플래그 값을 설정 (FLAG_SHOW_PERCENT).

```
void  ShowPercent(
  const bool  flag,  // 플래그 값
 \)
```

매개변수

Flags

[in] 플래그 값:

- true — 백분율이 표시됩니다.
- false — 백분율이 보이지 않게 됩니다.

© 2000-2025, MetaQuotes Ltd.

LegendAlignment

범례의 텍스트 정렬을 설정.

```
void  LegendAlignment(
  const ENUM_ALIGNMENT  value,  // 플래그
 \)
```

매개변수

값

[in] ENUM_ALIGNMENT 열거 값 중 하나를 가져오기:

- ALIGNMENT_LEFT — 좌측 정렬.
- ALIGNMENT_TOP — 위쪽 정렬.
- ALIGNMENT_RIGHT — 우측 정렬.
- ALIGNMENT_BOTTOM — 아래쪽 정렬.

© 2000-2025, MetaQuotes Ltd.

Accumulative

시리즈에 대한 값 누적 플래그를 설정.

```
void  Accumulative(
  const bool  flag=true,  // 플래그 값
 \)
```

매개변수

flag=true

[in] 플래그 값:

- true — 시리즈의 현재 값이 이전의 모든 값의 합으로 대체됩니다.
- false — 그리기 시리즈의 표준 모드.

© 2000-2025, MetaQuotes Ltd.

VScaleParams

값의 수직 스케일에 대한 매개변수를 설정.

```
void  VScaleParams(
  const double  max,   // 최대
  const double  min,   // 최소
  const uint    grid,  // 눈금 수
 \)
```

매개변수

max

[in] 최소값.

min

[in] 최대값.

그리드

[in] 스케일 눈금 수.

© 2000-2025, MetaQuotes Ltd.

DescriptorUpdate

지정된 포지션에 있는 시리즈 설명자 값 없데이트.

```
bool  DescriptorUpdate(
  const uint    pos,    // 인덱스
  const string  descr,  // 값
 \)
```

매개변수

pos

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

descr

[in] 설명자 값.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

ColorUpdate

지정된 포지션에서 시리즈 색상을 업데이트.

```
bool  ColorUpdate(
  const uint  pos,  // 인덱스
  const uint  clr,  // 색상
 \)
```

매개변수

pos

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

clr

[in] 색상 값.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

ValuesCheck

보조 가상 메서드, 차트를 표시하기 위한 내부 계산을 수행.

```
virtual void  ValuesCheck()
```

© 2000-2025, MetaQuotes Ltd.

Redraw

배경을 다시 그리는 가상 메서드.

```
virtual void  Redraw()
```

© 2000-2025, MetaQuotes Ltd.

DrawBackground

배경을 다시 그리는 가상 메서드.

```
virtual void  DrawBackground()
```

© 2000-2025, MetaQuotes Ltd.

DrawLegend

범례를 다시 그리는 가상 메서드.

```
virtual void  DrawLegend()
```

© 2000-2025, MetaQuotes Ltd.

DrawLegendVertical

수직 범례 그리기.

```
int  DrawLegendVertical(
  const int  w,  // 너비
  const int  h,  // 높이
 \)
```

매개변수

w

[in] 범례에서 텍스트의 최대 너비.

h

[in] 범례에서 텍스트의 최대 높이.

값 반환

범례의 너비(픽셀).

© 2000-2025, MetaQuotes Ltd.

DrawLegendHorizontal

수평 범례 그리기.

```
int  DrawLegendHorizontal(
  const int  w,  //
  const int  h,  //
 \)
```

매개변수

w

[in] 범례에서 텍스트의 최대 너비.

h

[in] 범례에서 텍스트의 최대 높이.

값 반환

범례의 높이(픽셀).

© 2000-2025, MetaQuotes Ltd.

CalcScales

값의 스케일에 대한 레이블의 좌표를 계산하는 가상 메서드.

```
virtual void  CalcScales()
```

© 2000-2025, MetaQuotes Ltd.

DrawScales

값의 모든 스케일을 다시 그리는 가상 메서드.

```
virtual void  DrawScales()
```

© 2000-2025, MetaQuotes Ltd.

DrawScaleLeft

스케일을 다시 그려야 하는지 여부를 나타내는 플래그.

```
virtual int  DrawScaleLeft(
  const bool  draw,  // 플래그
 \)
```

매개변수

그리기

[in] 스케일을 다시 그려야 하는지 여부를 나타내는 플래그.

값 반환

값 스케일의 너비.

© 2000-2025, MetaQuotes Ltd.

DrawScaleRight

우측 값 스케일을 다시 그리는 가상 메서드.

```
virtual int  DrawScaleRight(
  const bool  draw,  // 플래그
 \)
```

매개변수

그리기

[in] 스케일을 다시 그려야 하는지 여부를 나타내는 플래그.

값 반환

값 스케일의 너비.

© 2000-2025, MetaQuotes Ltd.

DrawScaleTop

값의 맨 위 스케일을 다시 그리는 가상 메서드.

```
virtual int  DrawScaleTop(
  const bool  draw,  // 플래그
 \)
```

매개변수

그리기

[in] 스케일을 다시 그려야 하는지 여부를 나타내는 플래그.

값 반환

값 스케일의 높이.

© 2000-2025, MetaQuotes Ltd.

DrawScaleBottom

값의 맨 아래 스케일을 다시 그리는 가상 메서드.

```
virtual int  DrawScaleBottom(
  const bool  draw,  // 플래그
 \)
```

매개변수

그리기

[in] 스케일을 다시 그려야 하는지 여부를 나타내는 플래그.

값 반환

값 스케일의 높이.

© 2000-2025, MetaQuotes Ltd.

DrawGrid

그리드를 다시 그리는 가상 메서드.

```
virtual void  DrawGrid()
```

© 2000-2025, MetaQuotes Ltd.

DrawChart

배경을 다시 그리는 가상 메서드.

```
virtual void  DrawChart()
```

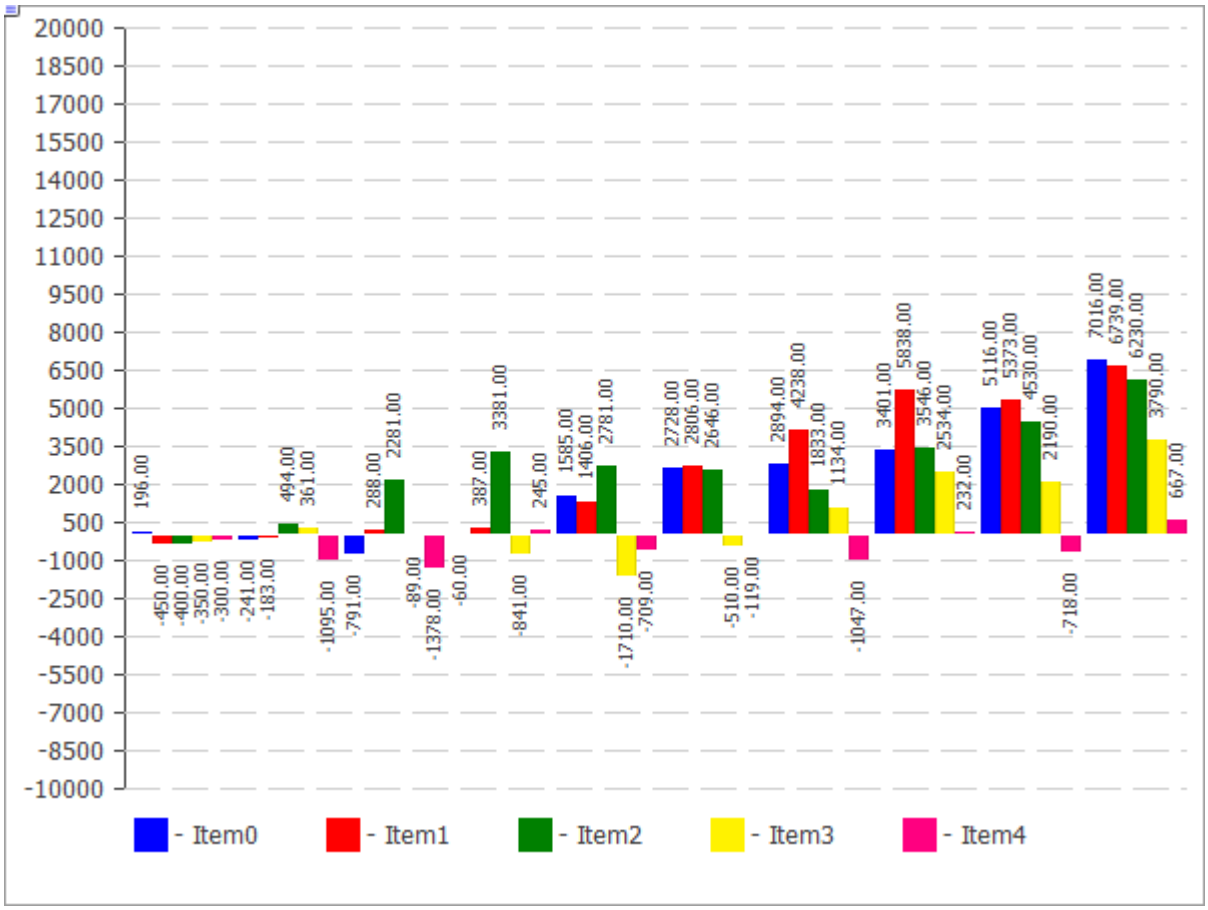
© 2000-2025, MetaQuotes Ltd.

CHistogramChart

히스토그램 플로팅하는 클래스.

설명

히스토그램의 플로팅으로 작업하는 모든 방법은 이 클래스에서 구현됩니다. 열 너비를 설정하고 데이터 시리즈로 작업을 구성하는 데 사용할 수 있습니다. 히스토그램 열의 그라데이션 채우기를 사용하여 데이터를 보다 명확하게 시각화 할 수 있는 방법이 포함되어 있습니다.



위 그림의 코드는 아래와 같습니다.

선언

```
class CHistogramChart : public CChartCanvas
```

제목

```
#include <Canvas\Charts\HistogramChart.mqh>
```

상속 계층

CCanvas
  CChartCanvas
    CHistogramChart

© 2000-2025, MetaQuotes Ltd.

클래스 메서드

메서드	액션
Gradient	히스토그램 열의 그라데이션 적용할지 여부를 나타내는 플래그를 설정.
BarGap	원점에서 히스토그램 오프셋 값을 설정합니다.
BarMinSize	히스토그램 열의 최소 너비를 설정.
BarBorder	각 열의 테두리를 그려야 함을 나타내는 플래그를 설정.
Create	그래픽 리소스를 생성하는 가상 메서드.
SeriesAdd	새 데이터 시리즈를 추가.
SeriesInsert	차으에 데이터 시리즈를 삽입.
SeriesUpdate	차트의 데이터 시리즈를 업데이트.
SeriesDelete	차트에서 데이터 시리즈를 삭제.
ValueUpdate	지정된 시리즈의 요소 값을 업데이트.
DrawData	지정된 시리즈에 대한 히스토그램을 표시하는 가상 방법.
DrawBar	히스토그램 열을 채워진 Rectangle으로 그리기.
GradientBrush	그라데이션 채우기에 사용할 브러시 생성.

클래스 CCanvas에서 상속된 메서드

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

CChartCanvas 클래스에서 상속된 메서드

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax,

© 2000–2025, MetaQuotes Ltd.

클래스 CCanvas에서 상속된 메서드

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

예

© 2000–2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                          HistogramChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "히스토그램 사용 예시"
//---
#include <Canvas\Charts\HistogramChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=true;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- 차트 생성
   CHistogramChart chart;
   if(!chart.CreateBitmapLabel("SampleHistogramChart",10,10,600,450))
     {
      Print("Error creating histogram chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,20);
   chart.ShowValue(true);
   chart.ShowScaleTop(false);
   chart.ShowScaleBottom(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- 값을 플레이
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 종료
   chart.Destroy();
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

Gradient

히스토그램 열의 그라데이션 적용할지 여부를 나타내는 플래그를 설정.

```
void  Gradient(
  const bool  flag=true,  // 플래그 값
 \)
```

매개변수

flag=true

플래그 값: 그라데이션 채우기가 활성화된 경우 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

BarGap

원점에서 히스토그램 오프셋 값을 설정합니다.

```
void  BarGap(
  const uint  value,  // 오프셋
 \)
```

매개변수

값

[in] 히스토그램 오프셋 값.

© 2000-2025, MetaQuotes Ltd.

BarMinSize

히스토그램 열의 최소 너비를 설정.

```
void  BarMinSize(
  const uint  value,  // 최소 너비
 \)
```

매개변수

값

[in] 최소 너비.

© 2000-2025, MetaQuotes Ltd.

BarBorder

각 열의 테두리를 그려야 함을 나타내는 플래그를 설정.

```
void  BarBorder(
  const uint  value,  // 플래그
 \)
```

매개변수

값

[in] 플래그 값:

- true — 테두리가 그려집니다
- false — 테두리가 그려지지 않습니다

© 2000-2025, MetaQuotes Ltd.

Create

그래픽 리소스를 생성하는 가상 메서드.

```
virtual bool  Create(
  const string       name,    // 이름
  const int          width,   // 너비
  const int          height,  // 높이
  ENUM_COLOR_FORMAT  clrfmt,  // 형식
 \)
```

매개변수

name

[in] 그래픽 리소스 이름의 기본. 생성 중에 의사 임의 문자열을 추가하여 리소스 이름을 생성합니다.

width

[in] 너비(X 축 방향 픽셀 크기).

height

[in] 높이(Y 축 방향 픽셀 크기).

clrfmt

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

SeriesAdd

새 데이터 시리즈를 추가.

```
bool  SeriesAdd(
  const double&  value[],  // 값
  const string   descr,    // 레이블
  const uint     clr,      // 색상
 \)
```

매개변수

value[]

[in] 데이터 시리즈.

descr

[in] 시리즈 레이블.

clr

[in] 시리즈 표시 색상.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

SeriesInsert

차으에 데이터 시리즈를 삽입.

```
bool  SeriesInsert(
  const uint    pos,        // 인덱스
  const double&  value[],   // 값
  const string  descr,      // 레이블
  const uint    clr,        // 색상
 \)
```

매개변수

pos

[in] 삽입용 인덱스.

value[]

[in] 데이터 시리즈.

descr

[in] 시리즈 레이블.

clr

[in] 시리즈 표시 색상.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

SeriesUpdate

Updates data series on the chart.

```
bool  SeriesUpdate(
  const uint    pos,        // 인덱스
  const double  &value[],   // 값
  const string  descr,      // 레이블
  const uint    clr,        // 색상
 \)
```

매개변수

pos

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

&value[]

[in] 데이터 시리즈의 새 값.

descr

[in] 시리즈 레이블.

clr

[in] 시리즈 표시 색상.

값 반환

성공하면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

SeriesDelete

차트에서 데이터 시리즈를 삭제.

```
bool  SeriesDelete(
  const uint  pos,  // 인덱스
 \)
```

매개변수

pos

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

지정된 시리즈에서 지정된 값을 업데이트.

```
bool  ValueUpdate(
  const uint   series,  // 시리즈의 인덱스
  const uint   pos,     // 요소의 인덱스
  double       value,   // 값
 \)
```

매개변수

시리즈

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

pos

[in] 시리즈의 요소 인덱스.

값

[in] 새 값.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

DrawData

지정된 시리즈에 대한 히스토그램을 표시하는 가상 방법.

```
virtual void  DrawData(
  const uint  index,  // 인덱스
 \)
```

매개변수

인덱스

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

© 2000-2025, MetaQuotes Ltd.

DrawBar

히스토그램 열을 채워진 Rectangle으로 그리기.

```
void  DrawBar(
  const int   x,    // X 좌표
  const int   y,    // Y 좌표
  const int   w,    // 너비
  const int   h,    // 높이
  const uint  clr,  // 색상
 \)
```

매개변수

x

[in] 다음에서의 좌측 상단 점의 X 좌표: Rectangle.

y

[in] 다음에서의 좌측 상단 점의 Y 좌표: Rectangle.

w

[in] Rectangle의 너비.

h

[in] Rectangle의 높이.

clr

[in] Rectangle의 색상.

© 2000-2025, MetaQuotes Ltd.

GradientBrush

그라데이션 채우기에 사용할 브러시 생성.

```
void  GradientBrush(
  const int   size,      // 크기
  const uint  fill_clr,  // 색상 채우기
 \)
```

매개변수

Size

[in] 브러시 두께

fill_clr

[in] 색상 채우기

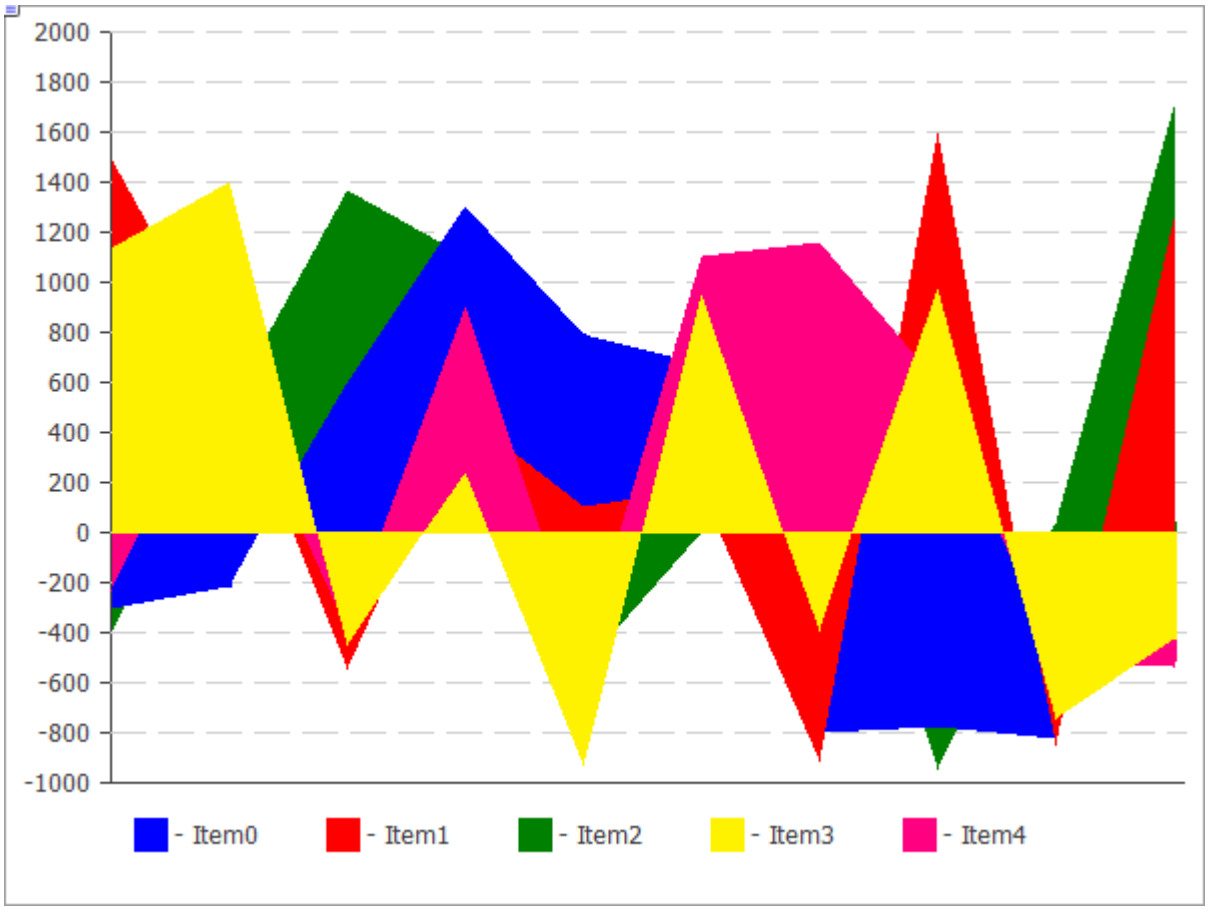
© 2000-2025, MetaQuotes Ltd.

CLineChart

곡선을 표시하는 클래스.

설명

이 클래스에 포함된 방법은 차트에서 곡선으로 작업할 수 있도록 설계되었습니다. 표시된 곡선으로 제한된 영역을 채울 수 있는 기능이 있습니다.



위 그림의 코드는 아래와 같습니다.

선언

```
class CLineChart : public CChartCanvas
```

제목

```
#include <Canvas\Charts\LineChart.mqh>
```

상속 계층

CCanvas
  CChartCanvas
    CLineChart

클래스 메서드

© 2000-2025, MetaQuotes Ltd.

메서드	액션
채워짐	데이터 시리즈에 의해 정의된 곡선 아래의 영역을 채우기 위한 플래그를 설정.
Create	그래픽 리소스 생성.
SeriesAdd	새 데이터 시리즈를 추가.
SeriesInsert	차으에 데이터 시리즈를 삽입.
SeriesUpdate	차트의 데이터 시리즈를 업데이트.
SeriesDelete	차트에서 데이터 시리즈를 삭제.
ValueUpdate	지정된 시리즈에서 지정된 값을 업데이트.
DrawChart	곡선과 그 모든 요소를 그리는 가상 메서드.
DrawData	지정된 열에 대해 곡선을 그리는 가상 메서드.
CalcArea	데이터 시리즈에 의해 정의된 곡선 아래의 면적을 계산.

클래스 CCanvas에서 상속된 메서드

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

CChartCanvas 클래스에서 상속된 메서드

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

예

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                LineChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
//|                                              http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "선 차트 사용의 예"
//---
#include <Canvas\Charts\LineChart.mqh>
//+------------------------------------------------------------------+
//| 입력                                                             |
//+------------------------------------------------------------------+
input bool Accumulative=false;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 차트 함수                                      |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- 차트 생성
   CLineChart chart;
//--- 차트 생성
   if(!chart.CreateBitmapLabel("SampleHistogrammChart",10,10,600,450))
     {
      Print("Error creating line chart: ",GetLastError());
      return(-1);
     }
   if(Accumulative)
     {
      chart.Accumulative();
      chart.VScaleParams(20*k*10,-10*k*10,20);
     }
   else
      chart.VScaleParams(20*k,-10*k,15);
   chart.ShowScaleTop(false);
   chart.ShowScaleRight(false);
   chart.ShowLegend();
   chart.Filled();
   for(int j=0;j<5;j++)
     {
      for(int i=0;i<10;i++)
        {
         k=-k;
         if(k>0)
            arr[i]=k*(i+10-j);
         else
            arr[i]=k*(i+10-j)/2;
        }
      chart.SeriesAdd(arr,"Item"+IntegerToString(j));
     }
//--- 값으로 플레이
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 종료
   chart.Destroy();
   return(0);
  }
```

© 2000-2025, MetaQuotes Ltd.

채워짐

데이터 시리즈에 의해 정의된 곡선 하단 영역을 채울 필요가 있는지 여부를 나타내는 플래그를 설정.

```
void  Filled(
  const bool  flag=true,  // 플래그
 \)
```

매개변수

flag=true

[in] 플래그 값:

- true — 커브 아래 영역을 채우기
- false —커브 아래 영역을 채우지 않음

© 2000-2025, MetaQuotes Ltd.

Create

그래픽 리소스를 생성하는 가상 메서드.

```
virtual bool  Create(
  const string       name,    // 이름
  const int          width,   // 너비
  const int          height,  // 높이
  ENUM_COLOR_FORMAT  clrfmt,  // 형식
 \)
```

매개변수

name

[in] 그래픽 리소스 이름의 기본. 생성 중에 의사 임의 문자열을 추가하여 리소스 이름을 생성합니다.

width

[in] 너비(X 축 방향 픽셀 크기).

height

[in] 높이(Y 축 방향 픽셀 크기).

clrfmt

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

SeriesAdd

새 데이터 시리즈를 추가.

```
bool  SeriesAdd(
  const double&  value[],  // 값
  const string   descr,    // 레이블
  const uint     clr,      // 색상
 \)
```

매개변수

value[]

[in] 데이터 시리즈.

descr

[in] 시리즈 레이블.

clr

[in] 시리즈 표시 색상.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

SeriesInsert

차으에 데이터 시리즈를 삽입.

```
bool  SeriesInsert(
  const uint     pos,        // 인덱스
  const double&  value[],    // 값
  const string   descr,      // 레이블
  const uint     clr,        // 색상
 \)
```

매개변수

pos

[in] 삽입용 인덱스.

value[]

[in] 데이터 시리즈.

descr

[in] 시리즈 레이블.

clr

[in] 시리즈 표시 색상.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

SeriesUpdate

Updates data series on the chart.

```
bool  SeriesUpdate(
  const uint     pos,        // 인덱스
  const double&  value[],    // 값
  const string   descr,      // 레이블
  const uint     clr,        // 색상
 \)
```

매개변수

pos

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

value[]

[in] 데이터 시리즈의 새 값.

descr

[in] 시리즈 레이블.

clr

[in] 시리즈 표시 색상.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

SeriesDelete

차트에서 데이터 시리즈를 삭제.

```
bool  SeriesDelete(
  const uint  pos,  // 인덱스
 \)
```

매개변수

pos

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

지정된 시리즈에서 지정된 값을 업데이트.

```
bool  ValueUpdate(
  const uint   series,  // 시리즈의 인덱스
  const uint   pos,     // 요소의 인덱스
  double       value,   // 값
 \)
```

매개변수

시리즈

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

pos

[in] 시리즈의 요소 인덱스.

값

[in] 새 값.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

DrawChart

곡선과 그 모든 요소를 그리는 가상 메서드.

```
virtual void  DrawChart()
```

© 2000-2025, MetaQuotes Ltd.

DrawData

지정된 열에 대해 곡선을 그리는 가상 메서드.

```
virtual void  DrawData(
  const uint  index,  // 인덱스
 \)
```

매개변수

인덱스

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

© 2000-2025, MetaQuotes Ltd.

CalcArea

데이터 시리즈에 의해 정의된 곡선 아래의 면적을 계산.

```
double  CalcArea(
  const uint  index,  // 인덱스
 \)
```

매개변수

인덱스

[in] 시리즈의 인덱스 —0으로 시작하는 추가의 일련 번호.

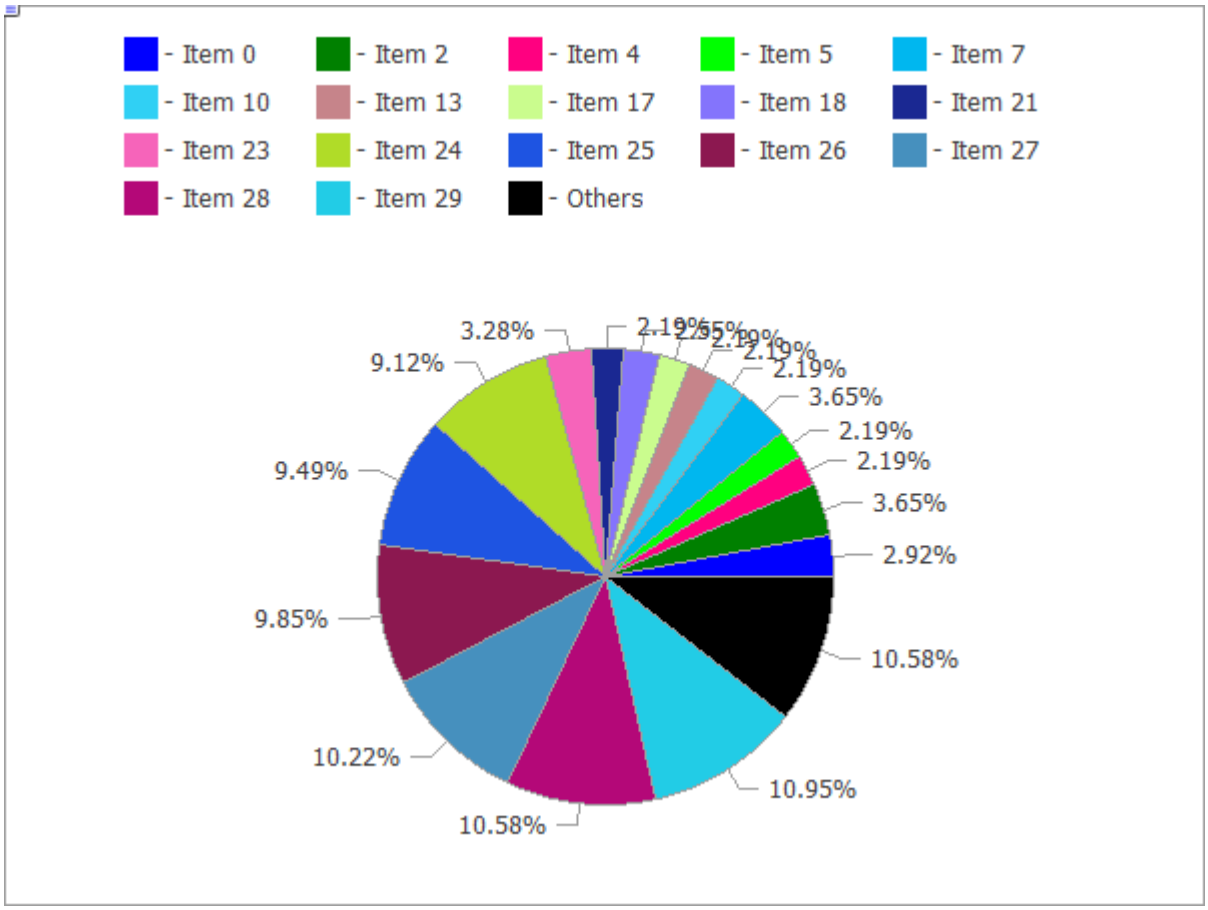
값 반환

데이터 시리즈에 의해 정의된 곡선에 의해 제한된 그림의 영역.

© 2000-2025, MetaQuotes Ltd.

CPieChart

파이 차트를 플로팅하는 클래스.



위 그림의 코드는 아래와 같습니다.

설명

이 클래스에 포함된 메서드는 그래픽 리소스 만들기부터 레이블 설계, 세그먼트 설계에 이르기까지 파이 차트를 사용하여 전체 작업을 수행할 수 있도록 설계되었습니다.

선언

```
class CPieChart : public CChartCanvas
```

제목

```
#include <Canvas\Charts\PieChart.mqh>
```

상속 계층

CCanvas
  CChartCanvas
    CPieChart

클래스 메서드

© 2000-2025, MetaQuotes Ltd.

メソド	액션
Create	그래픽 리소스를 생성하는 가상 메서드.
SeriesSet	파이 차트에 표시할 일련의 값을 설정.
ValueAdd	파이 차트의 끝에 새 값을 추가.
ValueInsert	지정한 포지션에 파이 차트에 새 값을 삽입.
ValueUpdate	지정된 포지션에 있는 파이 차트 값을 업데이트 .
ValueDelete	지정된 포지션에 있는 파이 차트에서 값을 제거 .
DrawChart	파이 차트와 해당 요소를 모두 그리는 가상 메서드.
DrawPie	지정된 값에 해당하는 파이 차트의 세그먼트 그리기.
LabelMake	값과 원래 레이블을 기준으로 세그먼트 레이블을 생성.

클래스 CCanvas에서 상속된 메서드

CreateBitmap, CreateBitmap, CreateBitmapLabel, CreateBitmapLabel, Attach, Attach, Destroy, ChartObjectName, ResourceName, Width, Height, Update, Resize, Erase, PixelGet, PixelSet, LineVertical, LineHorizontal, Line, Polyline, Polygon, Rectangle, Triangle, Circle, Ellipse, Arc, Arc, Arc, Pie, Pie, FillRectangle, FillTriangle, FillPolygon, FillCircle, FillEllipse, Fill, Fill, PixelSetAA, LineAA, PolylineAA, PolygonAA, TriangleAA, CircleAA, EllipseAA, LineWu, PolylineWu, PolygonWu, TriangleWu, CircleWu, EllipseWu, LineThickVertical, LineThickHorizontal, LineThick, PolylineThick, PolygonThick, PolylineSmooth, PolygonSmooth, FontSet, FontNameSet, FontSizeSet, FontFlagsSet, FontAngleSet, FontGet, FontNameGet, FontSizeGet, FontFlagsGet, FontAngleGet, TextOut, TextWidth, TextHeight, TextSize, GetDefaultColor, TransparentLevelSet, LoadFromFile, LineStyleGet, LineStyleSet

CChartCanvas 클래스에서 상속된 메서드

ColorBackground, ColorBackground, ColorBorder, ColorBorder, ColorText, ColorText, ColorGrid, ColorGrid, MaxData, MaxData, MaxDescrLen, MaxDescrLen, AllowedShowFlags, ShowFlags, ShowFlags, IsShowLegend, IsShowScaleLeft, IsShowScaleRight, IsShowScaleTop, IsShowScaleBottom, IsShowGrid, IsShowDescriptors, IsShowPercent, ShowLegend, ShowScaleLeft, ShowScaleRight, ShowScaleTop, ShowScaleBottom, ShowGrid, ShowDescriptors, ShowValue, ShowPercent, LegendAlignment, Accumulative, VScaleMin, VScaleMin, VScaleMax, VScaleMax, NumGrid, NumGrid, VScaleParams, DataOffset, DataOffset, DataTotal, DescriptorUpdate, ColorUpdate

예

```
//+------------------------------------------------------------------+
//|                                                PieChartSample.mq5 |
//|                  Copyright 2009-2017, MetaQuotes Software Corp. |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                                                 http://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2009-2017, MetaQuotes Software Corp."
#property link        "http://www.mql5.com"
#property description "Example of using pie chart"
//---
#include <Canvas\Charts\PieChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input int      Width=600;
input int      Height=450;
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
int OnStart(void)
  {
//--- check
   if(Width<=0 || Height<=0)
     {
      Print("Too simple.");
      return(-1);
     }
//--- 차트 생성
   CPieChart pie_chart;
   if(!pie_chart.CreateBitmapLabel("PieChart",10,10,Width,Height))
     {
      Print("파이 차트 생성 오류: ",GetLastError());
      return(-1);
     }
   pie_chart.ShowPercent();
//--- 그리기
   for(uint i=0;i<30;i++)
     {
      pie_chart.ValueAdd(100*(i+1),"Item "+IntegerToString(i));
      Sleep(10);
     }
   Sleep(2000);
//--- 범례 없애기
   pie_chart.LegendAlignment(ALIGNMENT_LEFT);
   Sleep(2000);
//--- 범례 없애기
   pie_chart.LegendAlignment(ALIGNMENT_RIGHT);
   Sleep(2000);
//--- 범례 없애기
   pie_chart.LegendAlignment(ALIGNMENT_TOP);
   Sleep(2000);
//--- 범례 없애기
   pie_chart.ShowLegend(false);
```

© 2000-2025, MetaQuotes Ltd.

```
   Sleep(2000);
//--- 백분율 없애기
   pie_chart.ShowPercent(false);
   Sleep(2000);
//--- 설명 없애기
   pie_chart.ShowDescriptors(false);
   Sleep(2000);
//--- 전체 표시
   pie_chart.ShowLegend();
   pie_chart.ShowValue();
   pie_chart.ShowDescriptors();
   Sleep(2000);
//--- 또는 이렇게
   pie_chart.ShowFlags(FLAG_SHOW_LEGEND|FLAG_SHOW_DESCRIPTORS|FLAG_SHOW_PERCENT);
   uint total=pie_chart.DataTotal();
//--- 값으로 플레이
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ValueUpdate(i,100*(rand()%10+1));
      Sleep(1000);
     }
//--- 색상으로 플레이
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ColorUpdate(i%total,RandomRGB());
      Sleep(1000);
     }
//--- 회전
   while(!IsStopped())
     {
      pie_chart.DataOffset(pie_chart.DataOffset()+1);
      Sleep(200);
     }
//--- 종료
   pie_chart.Destroy();
   return(0);
  }
//+------------------------------------------------------------------+
//| 랜덤 RGB 색상                                                    |
//+------------------------------------------------------------------+
uint RandomRGB(void)
  {
   return(XRGB(rand()%255,rand()%255,rand()%255));
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

그래픽 리소스를 생성하는 가상 메서드.

```
virtual bool  Create(
  const string       name,    // 이름
  const int          width,   // 너비
  const int          height,  // 높이
  ENUM_COLOR_FORMAT  clrfmt,  // 형식
 \)
```

매개변수

name

[in] 그래픽 리소스 이름의 기본. 생성 중에 의사 임의 문자열을 추가하여 리소스 이름을 생성합니다.

width

[in] 너비(X 축 방향 픽셀 크기).

height

[in] 높이(Y 축 방향 픽셀 크기).

clrfmt

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

SeriesSet

파이 차트에 표시할 일련의 값을 설정.

```
bool  SeriesSet(
  const double&  value[],  // 값
  const string&  text[],   // 레이블
  const uint&    clr[],    // 색상
 \)
```

매개변수

value[]

[in] 값의 배열.

text[]

[in] 값 레이블의 배열.

clr[]

[in] 값 색상의 배열.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

ValueAdd

파이 차트의 끝에 새 값을 추가.

```
bool  ValueAdd(
  const double  value,  // 값
  const string  descr,  // 레이블
  const uint    clr,    // 색상
 \)
```

매개변수

값

[in] 값.

descr

[in] 값 레이블.

clr

[in] 값 색상.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

ValueInsert

지정한 포지션에 파이 차트에 새 값을 삽입.

```
bool  ValueInsert(
  const uint    pos,    // 인덱스
  const double  value,  // 값
  const string  descr,  // 레이블
  const uint    clr,    // 색상
 \)
```

매개변수

pos

[in] 삽입용 인덱스.

값

[in] 값.

descr

[in] 값 레이블.

clr

[in] 값 색상.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

ValueUpdate

지정된 포지션에 있는 파이 차트 값을 업데이트.

```
bool  ValueUpdate(
  const uint    pos,    // 인덱스
  const double  value,  // 값
  const string  descr,  // 레이블
  const uint    clr,    // 색상
 \)
```

매개변수

pos

[in] 값 인덱스 —0으로 시작하는 추가의 일련 번호.

값

[in] 값.

descr

[in] 값 레이블.

clr

[in] 값 색상.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

ValueDelete

지정된 포지션에 있는 파이 차트에서 값을 제거.

```
bool  ValueDelete(
  const uint  pos,  // 인덱스
 \)
```

매개변수

pos

[in] 값 인덱스 —0으로 시작하는 추가의 일련 번호.

값 반환

성공하면 true, 그렇지 않은 경우 false.

© 2000-2025, MetaQuotes Ltd.

DrawChart

파이 차트와 해당 요소를 모두 그리는 가상 메서드.

```
virtual void  DrawChart()
```

© 2000-2025, MetaQuotes Ltd.

DrawPie

지정된 값에 해당하는 파이 차트의 세그먼트 그리기.

```
void  DrawPie(
  double       fi3,   // 호의 첫 번째 경계를 정의하는 파이 차트 중심으로부터의 광선 각도
  double       fi4,   // 호의 두 번째 경계를 정의하는 파트 차이 중심으로부터의 광선 각도
  int          idx,   // 인덱스
  CPoint&      p[],  //
  const uint   clr,   //
 \)
```

매개변수

fi3

[in] 호의 첫 번째 경계를 정의하는 라디안 단위의 각도.

fi4

[in] 호의 두 번째 경계를 정의하는 라디안 단위의 각도.

idx

[in] 세그먼트가 해당하는 값의 인덱스.

p[]

[in] 세그먼트를 표시하기 위한 기준점 배열 (x, y).

clr

[in] 세그먼트의 색상.

© 2000-2025, MetaQuotes Ltd.

LabelMake

값과 원래 레이블을 기준으로 세그먼트 레이블을 생성.

```
string  LabelMake(
  const string  text,     // 레이블
  const double  value,    // 값
  const bool    to_left,  // 플래그
 \)
```

매개변수

텍스트

[in] 레이블.

값

[in] 값.

to_left

[in] 레이블 레이아웃 순서를 정의:

- true — 레이블, 다음 값.
- false — 값, 다음 레이블.

값 반환

세그먼트의 레이블.

© 2000-2025, MetaQuotes Ltd.

3D 그래픽

이 섹션에서는 3D 그래픽 개발을 위한 클래스를 제공합니다. 클래스는 DirectX를 사용하기 위한 기능을 기반으로 합니다. CCnavas3D는 카메라와 조명을 관리하는 방법과 텍스처, 셰이더, 정점 버퍼, 인덱스 및 셰이더 매개변수의 관리자를 포함하는 기본 클래스입니다.

라이브러리 작업을 시작하려면 MetaTrader 5에서 DirectX를 사용하여 3D 그래픽 생성 방법 아티클을 읽으십시오.

또한 상자, 사용자 데이터의 3D 표면 및 임의 그리드와 같은 장면(scene) 기본 개체 클래스가 있습니다.

© 2000-2025, MetaQuotes Ltd.

CCanvas3D

CCanvas3D는 차트에서 3D 개체를 간단하게 만들고 시각화하는 클래스입니다.

설명

CCanvas3D는 애니메이션 3D 그래픽 형태로 대량 데이터를 만들고 시각화하는 작업을 크게 간소화합니다. 클래스에는 카메라 및 조명을 관리하는 방법과 그래픽 리소스(예: 텍스처, 셰이더, 정점 버퍼, 인덱스 및 셰이더 매개변수)를 작성하는 리소스 관리자가 포함되어 있습니다.

또한 라이브러리에는 상자, 사용자 데이터의 3D 표면 또는 임의 그리드와 같은 기본 개체 클래스가 포함되어 있습니다.

기능이 작동하려면 비디오 카드가 DX 11 및 셰이더 모델 5.0을 지원해야 합니다.

선언

```
class CCanvas
```

제목

```
#include <Canvas\Canvas.mqh>
```

상속 계층

CCanvas

CCanvas3D

그룹별 클래스 메서드

생성/삭제	설명
Create	차트 개체에 바인딩하지 않고 3D 장면을 렌더링하기 위한 그래픽 리소스를 작성합니다.
Attach	OBJ_BITMAP_LABEL 개체에서 그래픽 리소스를 가져와서 CCanvas 클래스 인스턴스에 연결합니다.
ObjectAdd	후속 렌더링을 위해 3D 장면에 개체를 추가합니다.
Delete	그래픽 리소스를 삭제하고 3D 컨텍스트를 릴리스합니다.
조명	
AmbientColorSet	주변 전체 라이팅의 색상과 강도를 설정합니다.
AmbientColorGet	주변 전체 라이팅의 색상과 강도를 가져옵니다
LightDirectionSet	방향 지정 광원의 방향을 설정합니다.
LightDirectionGet	방향 지정 광원의 방향을 가져옵니다.
LightColorSet	방향이 있는 광원의 색상과 강도를 설정합니다.
LightColorGet	방향이 있는 광원의 색상과 강도를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

생성/삭제	설명
Camera	
ProjectionMatrixSet	3D 좌표 투영 매트릭스를 계산하여 2D 프레임으로 설정합니다.
ProjectionMatrixGet	3D 장면 투영 매트릭스를 2D 프레임으로 가져옵니다.
ViewMatrixSet	3D 장면 뷰 매트릭스를 설정합니다.
ViewMatrixGet	3D 장면 뷰 매트릭스를 반환합니다.
ViewPositionSet	3D 장면의 시점을 설정합니다.
ViewRotationSet	3D 장면에서 시선 방향을 설정합니다.
ViewTargetSet	시선이 향하는 지점의 좌표를 설정합니다.
ViewUpDirectionSet	3D 공간에서 상부 프레임 테두리의 방향을 설정합니다.
렌더링	
Render	이후 디스플레이를 위해 프레임 내부 버퍼의 모든 장면 개체를 렌더링합니다.
RenderBegin	새 프레임을 렌더링하기 위한 그래픽 컨텍스트를 준비합니다.
RenderEnd	렌더링된 프레임을 내부 버퍼에 복사하고 필요한 경우 차트 이미지를 업데이트합니다.
리소스 가져오기	
DXContext	그래픽 컨텍스트 핸들을 가져옵니다.
DXDispatcher	리소스 발송자 핸들을 가져옵니다.
InputScene	장면 매개변수의 버퍼에 대한 포인터를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

AmbientColorGet

주변 전체 라이팅의 색상과 강도를 가져옵니다

```
void  AmbientColorGet(
   DXColor  &ambient_color      // 전면 라이팅 색상과 강도
   );
```

매개변수

&ambient_color

[out] 전면 라이팅 색상.

값 반환

아니오

참고

명암은 DXColor 구조의 알파 채널에 저장됩니다.

© 2000-2025, MetaQuotes Ltd.

AmbientColorSet

주변 전체 라이팅의 색상과 강도를 설정합니다.

```
void  AmbientColorSet(
   const DXColor  &ambient_color      // 전면 라이팅 색상과 강도
   );
```

매개변수

&ambient_color

[in] 전면 라이팅 색상.

값 반환

아니오

참고

명암은 DXColor 구조의 알파 채널에서 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

Attach

OBJ_BITMAP_LABEL 개체에서 그래픽 리소스를 가져와 CCanvas 클래스의 인스턴스에 연결합니다.

```
bool  Attach(
   const long          chart_id,                          // 차트 ID
   const string        objname,                           // 개체 이름
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // 색상 처리 방법
  \)
```

리소스(OBJ_BITMAP_LABEL 개체에 대한)를 만들고 이를 CCanvas 클래스의 인스턴스에 연결합니다.

```
bool  Attach(
   const long          chart_id,                          // 차트 ID
   const string        objname,                           // 개체 이름
   const int           width,                             // 이미지 너비(픽셀)
   const int           height,                            // 이미지 높이(픽셀)
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // 색상 처리 방법
  \)
```

매개변수

chart_id

[in] 차트 ID.

objname

[in] 그래픽 개체의 이름.

width

[in] 리소스의 프레임 너비.

height

[in] 프레임 높이.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

참고

그래픽 개체 추가에 성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

Create

차트 개체에 바인딩하지 않고 3D 장면을 렌더링하기 위한 그래픽 리소스를 작성합니다.

```
virtual bool  Create(
   const string       name,                                  // 그래픽 개체 이름
   const int          width,                                 // 너비
   const int          height,                                // 높이
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // 색상 형식
   );
```

매개변수

name

[in] 그래픽 개체 이름.

width

[in] 프레임 너비.

height

[in] 프레임 높이.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] 색상 처리 방법. 색상 처리 방법에 대한 자세한 내용은 ResourceCreate() 함수 설명을 참조하십시오.

참고

생성에 성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

Delete

그래픽 리소스를 삭제하고 3D 컨텍스트를 릴리스합니다.

```
virtual void  Destroy()
```

값 반환

아니오

참고

그래픽 리소스가 차트 개체에 바인딩된 경우 후자가 삭제됩니다.

© 2000-2025, MetaQuotes Ltd.

DXContext

그래픽 컨텍스트 핸들을 가져옵니다.

```
int  DXContext()
```

값 반환

그래픽 컨텍스트 핸들.

© 2000-2025, MetaQuotes Ltd.

DXDispatcher

리소스 발송자 핸들을 가져옵니다.

```
CDXDispatcher*  DXDispatcher()
```

값 반환

리소스 발송자 핸들.

© 2000-2025, MetaQuotes Ltd.

InputScene

장면 매개변수의 버퍼에 대한 포인터를 가져옵니다.

```
CDXInput*  InputScene()
```

값 반환

장면 매개변수의 버ㅍ퍼에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

LightColorGet

방향이 있는 광원의 색상과 강도를 가져옵니다.

```
void  LightColorGet(
   DXColor  &light_color      // 방향성을 가진 라이팅 색상과 강도
   );
```

매개변수

&light_color

[out] 방향성을 가진 라이팅 색상과 강도.

값 반환

없음.

참고

명암은 DXColor 구조의 알파 채널에 저장됩니다.

© 2000-2025, MetaQuotes Ltd.

LightColorSet

방향이 있는 광원의 색상과 강도를 설정합니다.

```
void  LightColorSet(
   const DXColor  &light_color      // 방향성을 가진 라이팅 색상과 강도
   );
```

매개변수

&light_color

[in] 방향성을 가진 라이팅 색상과 강도.

값 반환

없음.

참고

명암은 DXColor 구조의 알파 채널에서 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

LightDirectionGet

방향 지정 광원의 방향을 가져옵니다.

```
void  LightDirectionGet(
   DXVector3  &light_direction      // 방향 벡터
   );
```

매개변수

&light_direction

[out] 방향 벡터.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

LightDirectionSet

방향 지정 광원의 방향을 설정합니다.

```
void  LightDirectionSet(
   const DXVector3  &light_direction      // 방향 벡터
   );
```

매개변수

&light_direction

[in] 방향 벡터.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

ObjectAdd

후속 렌더링을 위해 3D 장면에 개체를 추가합니다.

```
bool  ObjectAdd(
   CDXObject  *object      // 개체에 대한 포인터
   );
```

매개변수

object

[in] CDXObject 추상 클래스에서 파생된 클래스의 인스턴스에 대한 포인터.

값 반환

3D 그래픽 개체를 추가하는데 성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

ProjectionMatrixGet

3D 장면 투영 매트릭스를 2D 프레임으로 가져옵니다.

```
void  ProjectionMatrixGet(
   DXMatrix  &projection_matrix      // 투영 매트릭스
   );
```

매개변수

&projection_matrix

[out] 투영 매트릭스.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

ProjectionMatrixSet

3D 좌표 투영 매트릭스를 계산하여 2D 프레임으로 설정합니다.

```
void  ProjectionMatrixSet(
   float  fov,              // 시야
   float  aspect_ratio,     // 프레임 가로 세로 비율
   float  z_near,           //
   float  z_far             //
   );
```

매개변수

fov

[in] 장면 투영을 작성하기 위한 가시 폭 필드(라디안).

aspect_ratio

[in] 2D 프레임 가로 세로 비율.

z_near

[in] 근접 클리핑 평면까지의 거리.

z_far

[in] 근접 클리핑 평면까지의 거리.

값 반환

없음.

참고

2D 프레임은 지정된 시야에 들어가 근거리 및 원거리 클리핑 평면 사이에 위치한 3D 개체의 투명만 표시합니다.

© 2000-2025, MetaQuotes Ltd.

Render

이후 디스플레이를 위해 프레임 내부 버퍼의 모든 장면 개체를 렌더링합니다.

```
bool  Render(
   uint  flags,                  // 플래그 조합
   uint  background_color=0      // 배경 색상
   );
```

매개변수

flags

[in] 렌더링 모드를 설정하는 플래그 조합. 가능한 값:
DX_CLEAR_COLOR - *background_color*를 사용하여 이미지 버퍼를 지우기.
DX_CLEAR_DEPTH - 깊이 버퍼 지우기.

background_color=0

[in] 3D 장면 배경 색.

값 반환

렌더링에 성공하면 true, 실패하면 false.

참고

Render()를 호출해도 차트의 장면이 업데이트되지는 않습니다. 대신 이미지의 내부 버퍼만 업데이트합니다. 업데이트된 프레임을 렌더링하려면 Update() 메서드를 명시적으로 오출해야 합니다.

Render()는 RenderBegin 및 RenderEnd() 호출하는 기능입니다.

© 2000-2025, MetaQuotes Ltd.

RenderBegin

새 프레임을 렌더링하기 위한 그래픽 컨텍스트를 준비합니다.

```
virtual bool  RenderBegin(
   uint  flags,                  // 플래그 조합
   uint  background_color=0      // 배경 색상
   );
```

매개변수

flags

[in] 렌더링 모드를 설정하는 플래그 조합입니다. 가능한 값:
DX_CLEAR_COLOR - *background_color*를 사용하여 이미지 버퍼를 지우기.
DX_CLEAR_DEPTH – 깊이 버퍼 지우기.

background_color=0

[in] 3D 장면 배경 색.

값 반환

셰이더 입력을 업데이트하는데 성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

RenderEnd

렌더링된 프레임을 내부 버퍼에 복사하고 필요한 경우 차트 이미지를 업데이트합니다.

```
virtual bool  RenderEnd(
   bool  redraw=false      // 업데이트 플래그
   );
```

매개변수

redraw=false

[in] 필요성을 다시 그리는 차트 플래그.

값 반환

성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

ViewMatrixGet

3D 장면 뷰 매트릭스를 반환합니다.

```
void  ViewMatrixGet(
   DXMatrix  &view_matrix      // 뷰 매트릭스
   );
```

매개변수

&view_matrix

[out] 3D 공간에서 카메라 위치와 방향을 설정하는 매트릭스를 봅니다.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

ViewMatrixSet

3D 장면 뷰 매트릭스를 설정합니다.

```
void  ViewMatrixSet(
   const DXMatrix  &view_matrix      // 뷰 매트릭스
   );
```

매개변수

&view_matrix

[in] 3D 공간에서 카메라 위치와 방향을 설정하는 매트릭스를 봅니다.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

ViewPositionSet

3D 장면의 시점을 설정합니다.

```
void  ViewPositionSet(
   const DXVector3  &position      // 시점 위치
   );
```

매개변수

&position

[in] 3D 장면의 시점을 설정.

값 반환

없음.

참고

ViewPositionSet()를 사용하여 시점 위치를 설정하면 ViewMatrixGet()에서 얻은 뷰 매트릭스가 변경됩니다.

© 2000-2025, MetaQuotes Ltd.

ViewRotationSet

3D 장면에서 시선 방향을 설정합니다.

```
void  ViewRotationSet(
   const DXVector3  &rotation      // 회전각의 백터
   );
```

매개변수

&rotation

[in] 벡터 설정 오일러 각도를 사용하여 3D 장면에서 시선 방향을 계산.

값 반환

없음.

참고

ViewRotationSet()를 사용하여 시선 방향을 설정하면ViewMatrixGet()에서 얻은 뷰 매트릭스가 변경됩니다.

© 2000-2025, MetaQuotes Ltd.

ViewTargetSet

시선이 향하는 지점의 좌표를 설정합니다.

```
void  ViewTargetSet(
   const DXVector3  &target      // 타겟 좌표
   );
```

매개변수

&target

[in] 시선이 향하는 지점의 좌표.

값 반환

없음.

참고

시점을 이동할 때 한 장면 지점에서 시선을 고정하는 데 사용됩니다.

ViewRotationSet()를 사용하여 새 타겟 좌표를 설정하면 ViewMatrixGet()에서 얻은 뷰 매트릭스가 변경됩니다.

ViewTargetSet()을 ViewUpDirectionSet()와 함께 사용하여 시선 방향을 정의합니다.

© 2000-2025, MetaQuotes Ltd.

ViewUpDirectionSet

3D 공간에서 상부 프레임 테두리의 방향을 설정합니다.

```
void  ViewUpDirectionSet(
   const DXVector3  &up_direction      // 위 방향
   );
```

매개변수

&up_direction

[in] 3D 공간의 프레임 위 방향입니다.

값 반환

없음.

참고

ViewUpDirectionSet()를 사용하여 새 방향을 설정하면 ViewMatrixGet()에서 얻은 뷰 매트릭스가 변경됩니다.

ViewUpDirectionSet()는 ViewTargetSet()와 함께 시선 방향을 정의하는 데 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

CChart

CChart는 "차트" 그래픽 개체 속성에 대한 단순 액세스를 위한 클래스입니다.

설명

CChart 클래스는 "차트" 개체 속성에 대한 액세스를 제공합니다.

선언

```
class CChart : public CObject
```

제목

```
#include <Charts\Chart.mqh>
```

상속 계층

CObject
  CChart

그룹별 클래스 메서드

보호된 데이터에 대한 액세스	
ChartID	차트의 식별자를 가져오기
일단 속성	
Mode	"모드" 속성 값(막대, 캔들 또는 선)을 가져오기/설정하기
Foreground	"전경" 속성 값 가져오기/설정하기
Shift	"변화" 속성 값 가져오기/설정하기
ShiftSize	"ShiftSize" 속성 값(%)을 가져오기/설정하기
AutoScroll	"AutoScroll" 속성 값 가져오기/설정하기
Scale	"스케일" 속성 값 가져오기/설정하기
ScaleFix	"ScaleFix" 속성 값(고정 차트 스케일 여부)을 가져오기/설정하기
ScaleFix_11	"ScaleFix_11" 속성 값(차트 스케일 1:1 여부)을 가져오기/설정하기
FixedMax	"FixedMax" 속성 값(고정 최대 가격)을 가져오기/설정하기
FixedMin	"FixedMin" 속성 값(최소 고정 가격)을 가져오기/설정하기
ScalePPB	"ScalePPB" 속성 값(스케일 "막대당 포인트" 여부)을 가져오기/설정하기
PointsPerBar	"PointsPerBar" 속성 값(막대당 포인트)을 가져오기/설정하기
속성 표시	

© 2000-2025, MetaQuotes Ltd.

보호된 데이터에 대한 액세스	
ShowOHLC	"ShowOHLC" 속성 값을 가져오기/설정하기
ShowLineBid	"ShowLineBid" 속성 값을 가져오기/설정하기
ShowLineAsk	"ShowLineAsk" 속성 값을 가져오기/설정하기
ShowLastLine	"ShowLastLine" 속성 값을 가져오기/설정하기
ShowPeriodSep	"ShowPeriodSep" 속성 값(기간 구분자 표시)을 가져오기/설정하기
ShowGrid	"ShowGrid" 속성 값을 가져오기/설정하기
ShowVolumes	"ShowVolumes" 속성 값(볼륨 및 열린 포지션 수준의 색상)을 가져오기/설정하기
ShowObjectDescr	"ShowObjectDescr" 속성 값(그래픽 개체에 대한 설명 표시)을 가져오기/설정하기
ShowDateScale	"ShowDateScale" 속성 값(차트의 날짜 스케일) 설정
ShowPriceScale	"ShowPriceScale" 속성 값(차트의 가격 스케일) 설정
색상 설정	
ColorBackground	"ColorBackground" 속성 값(차트의 배경색)을 가져오기/설정하기
ColorForeground	"ColorForeground" 속성 값(차트의 축, 스케일 및 OHLC 문자열 색상)을 가져오기/설정하기
ColorGrid	"ColorGrid" 속성 값(그리드 색상)을 가져오기/설정하기
ColorBarUp	"ColorBarUp" 속성 값(상승 막대의 색상, 그림자 및 캔들 본체 윤곽선)을 가져오기/설정하기
ColorBarDown	"ColorBarDown" 속성 값(하락 막대의 색상, 그림자 및 캔들 본체의 윤곽선)을 가져오기/설정하기
ColorCandleBull	"ColorCandleBull" 속성 값(상승 캔들 본체 색상)을 가져오기/설정하기
ColorCandleBear	"ColorCandleBear" 속성 값(하락 캔들 본체 색상)을 가져오기/설정하기
ColorChartLine	"ColorChartLine" 속성 값(선형 차트 및 도지 캔들의 색상)을 가져오기/설정하기
ColorVolumes	"ColorVolumes" 속성 값(볼륨 및 열린 포지션의 수준 색상)을 가져오기/설정하기
ColorLineBid	"ColorLineBid" 속성 값(입찰 선 색상)을 가져오기/설정하기
ColorLineAsk	"ColorLineAsk" 속성 값(호가 선 색상)을 가져오기/설정하기
ColorLineLast	"ColorLineLast" 속성 값(마지막 거래 가격대의 색상)을 가져오기/설정하기

© 2000-2025, MetaQuotes Ltd.

보호된 데이터에 대한 액세스	
ColorStopLevels	"ColorStopLevels" 속성 값(SL 및 TP 레벨의 색상)을 가져오기/설정하기
읽기 전용 속성	
VisibleBars	표시되는 차트 막대의 총 수 가져오기
WindowsTotal	차트 지표 하위 창을 포함하여 총 차트 창 수를 가져오기
WindowIsVisible	지정된 차트 하위 창의 가시성 플래그를 가져오기
WindowHandle	차트의 창 핸들을 가져오기(HWND)
FirstVisibleBar	차트의 첫 번째 표시 막대 수를 가져오기
WidthInBars	창 너비(막대) 가져오기.
WidthInPixels	하위 창 너비(픽셀) 가져오기.
HeightInPixels	하위 창 높이(픽셀) 가져오기.
PriceMin	지정된 하위 창 최소 가격 가져오기
PriceMax	지정된 하위 창 최대 가격 가져오기
속성	
Attach	현재 차트를 클래스 인스턴스에 할당하기
FirstChart	클라이언트 터미널의 첫 번째 차트를 클래스 인스턴스에 할당하기
NextChart	클라이언트 터미널의 다음 차트를 클래스 인스턴스에 할당하기
Open	지정된 매개변수가 있는 차트를 열고 클래스 인스턴스에 할당하기
Detach	차트를 클래스 인스턴스에서 분리
Close	클래스 인스턴스에 할당된 차트를 닫기
BringToTop	다른 차트 위에 차트 표시
EventObjectCreate	차트에 새 개체 생성 이벤트 알림을 보내도록 플래그를 설정
EventObjectDelete	차트에서 개체 삭제 이벤트에 대한 알림을 보내도록 플래그 설정
지표	
IndicatorAdd	지정된 핸들을 가진 지표를 지정된 차트 하위 창에 추가
IndicatorDelete	지정된 차트 하위 창에서 지정된 이름의 지표를 제거
IndicatorsTotal	지정된 차트 하위 창에 적용된 모든 지표 수를 반환
IndicatorName	지정된 차트 하위 창에 있는 지표의 단축명을 반환
네비게이션	

© 2000-2025, MetaQuotes Ltd.

보호된 데이터에 대한 액세스	
Navigate	차트를 탐색
MQL5 API에 액세스	
Symbol	차트의 기호 가져오기
Period	차트의 주기 가져오기
Redraw	클래스 인스턴스에 할당된 차트 다시 그리기
GetInteger	함수는 해당 개체 속성의 값을 반환
SetInteger	정수 유형의 속성에 대한 새 값을 설정
GetDouble	함수는 해당 개체 속성의 값을 반환
SetDouble	double 유형의 속성에 대한 새 값을 설정
GetString	함수는 해당 개체 속성의 값을 반환
SetString	string 유형의 속성에 대한 새 값을 설정
SetSymbolPeriod	클래스 인스턴스에 할당된 차트의 기호 및 기간을 변경
ApplyTemplate	차트에 지정된 템플릿을 적용
ScreenShot	지정된 차트의 스크린 샷을 만들어 .gif 파일에 저장
WindowOnDropped	개체(전문가 또는 스크립트) 드롭 포인트에 해당 차트 하위 창 번호를 가져오기
PriceOnDropped	개체(전문가 또는 스크립트)의 하락점에 해당하는 가격 좌표 가져오기
TimeOnDropped	개체(전문가 또는 스크립트) 드롭 포인트에 해당하는 시간 좌표 가져오기
XOnDropped	개체(전문가 또는 스크립트) 드롭 포인트에 해당하는 X 좌표를 가져오기
YOnDropped	개체(전문가 또는 스크립트) 드롭 포인트에 해당하는 Y 좌표를 가져오기
입력/출력	
virtual Save	개체 매개변수를 파일에 저장
virtual Load	파일에서 개체 매개변수를 로드
virtual Type	그래픽 개체 유형 식별자 가져오기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

© 2000-2025, MetaQuotes Ltd.

ChartID

차트의 식별자 반환.

```
long  ChartID() const
```

값 반환

차트 클래스 인스턴스에 할당된 차트 식별자. 할당된 차트가 없으면 -1을 반환.

© 2000-2025, MetaQuotes Ltd.

Mode (Get Method)

"모드" 속성 값(막대, 캔들 또는 선)을 가져오기.

```
ENUM_CHART_MODE  Mode() const
```

값 반환

클래스 인스턴스에 할당된 개체의 "모드" 속성 값입니다. 차트가 할당되지 않은 경우 WRONG_VALUE를 반환합니다.

Mode (Set Method)

"모드" 속성(막대, 캔들 또는 선)에 대한 새 값을 설정.

```
bool  Mode(
   ENUM_CHART_MODE  mode      // 차트 모드
  \)
```

매개변수

mode

[in] ENUM_CHART_MODE 열거의 차트 모드(캔들, 막대 또는 선).

값 반환

성공하면 true, 모드를 변경하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

Foreground (Get Method)

"전경" 속성 값을 가져오기.

```
bool  Foreground() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "전경" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

Foreground (Set Method)

"전경" 속성의 새 값을 설정.

```
bool  Foreground(
   bool  foreground      // 플래그 값
  \)
```

매개변수

foreground

[in] "전경" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

Shift (Get Method)

"시프트" 속성 값 가져오기.

```
bool  Shift() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "시프트" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

Shift (Set Method)

"시프트" 속성 새 값 설정.

```
bool  Shift(
   bool  shift      // 플래그 값
  \)
```

매개변수

shift

[in] "시프트" 속성 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShiftSize (Get Method)

"ShiftSize" 속성 값(%)을 가져오기.

```
double  ShiftSize() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ShiftSize" 속성 값. 차트가 할당되지 않은 경우 EMPTY_VALUE를 반환합니다.

ShiftSize (Set Method)

"시프트" 속성의 새 값(%)을 설정.

```
bool  ShiftSize(
   double  shift_size      // 속성 값
  \)
```

매개변수

shift_size

[in] "ShiftSize" 속성의 새 값(%).

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

AutoScroll (Get Method)

"AutoScroll" 속성 값 가져오기.

```
bool  AutoScroll() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "AutoScroll" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

AutoScroll (Set Method)

"AutoScroll" 속성의 새 값을 설정.

```
bool  AutoScroll(
   bool  autoscroll      // 플래그 값
  \)
```

매개변수

autoscroll

[in] "Autoscroll" 속성의 새로운 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

Scale (Get Method)

"스케일" 속성 값을 가져오기.

```
int  Scale() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "스케일" 속성 값. 할당된 차트가 없으면 0을 반환합니다.

Scale (Set Method)

"스케일" 속성 새 값 설정하기.

```
bool  Scale(
   int  scale      // 속성 값
  \)
```

매개변수

scale

[in] "스케일" 속성 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ScaleFix (Get Method)

"ScaleFix" 속성 값(고정 차트 스케일 여부)을 가져오기.

```
bool  ScaleFix() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ScaleFix" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

ScaleFix (Set Method)

"ScaleFix" 속성의 새 값을 설정.

```
bool  ScaleFix(
   bool  scale_fix      // 속성 값
  \)
```

매개변수

scale_fix

[in] "ScaleFix" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ScaleFix_11 (Get Method)

"ScaleFix_11" 속성 값(차트 스케일은 1:1 여부)을 가져오기.

```
bool  ScaleFix_11() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ScaleFix_11" 속성 값. 할당된 차트가 없으면, false를 반환합니다.

ScaleFix_11 (Set Method)

"ScaleFix_11" 속성의 새 값을 설정.

```
bool  ScaleFix_11(
   string  scale_11      // 속성 값
  \)
```

매개변수

scale_11

[in] "ScaleFix_11" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

FixedMax (Get Method)

"FixedMax" 속성 값(고정 최대 가격)을 가져오기.

```
double  FixedMax() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "FixedMax" 속성 값. 차트가 할당되지 않은 경우 EMPTY_VALUE를 반환합니다.

FixedMax (Set Method)

"FixedMax" 속성의 새 값을 설정.

```
bool  FixedMax(
   double  max      // 고정 최대
  \)
```

매개변수

max

[in] "FixedMax" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

FixedMin (Get Method)

"FixedMin" 속성 값(최소 고정 가격)을 가져오기.

```
double  FixedMin() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "FixedMin" 속성 값. 차트가 할당되지 않은 경우 EMPTY_VALUE를 반환합니다.

FixedMin (Set Method)

"FixedMin" 속성의 새 값을 설정.

```
bool  FixedMax(
   double  min      // 고정 최소
  \)
```

매개변수

max

[in] "FixedMin" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

PointsPerBar (Get Method)

"PointsPerBar" 속성 값(막대당 포인트)을 가져옵니다.

```
double  PointsPerBar() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "PointsPerBar" 속성 값. 할당된 차트가 없으면, EMPTY_VALUE를 반환합니다.

PointsPerBar (Set Method)

"PointsPerBar" 속성 새 값 설정.

```
bool  PointsPerBar(
   double  ppb      // 스케일
  \)
```

매개변수

ppb

[in] 새 스케일 (막대당 포인트).

값 반환

성공하면 true, 스케일을 변경하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ScalePPB (Get Method)

"ScalePPB" 속성 값(스케일이 "막대당 포인트"인지 여부)을 가져오기.

```
bool  ScalePPB() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ScalePPB" 속성 값. 할당된 차트가 없으면, false를 반환합니다.

ScalePPB (Set Method)

"ScalePPB" 속성의 새 값 설정.

```
bool  ScalePPB(
   bool  scale_ppb      // 플래그 값
  \)
```

매개변수

scale_ppb

[in] "ScalePPB" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowOHLC (Get Method)

"ShowOHLC" 속성 값 가져오기.

```
bool  ShowOHLC() const
```

값 반환

클래스 인스턴스에 할당된 타츠의 "ShowOHLC" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

ShowOHLC (Set Method)

"ShowOHLC" 속성의 새 값 설정.

```
bool  ShowOHLC(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowOHLC" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ShowLineBid (Get Method)

"ShowLineBid" 속성의 값 가져오기.

```
bool  ShowLineBid() const
```

값 반환

클래스 인스턴스에 할당된 char의 "ShowLineBid" 속성 값 . 할당된 차트가 없으면 false를 반환합니다.

ShowLineBid (Set Method)

"ShowLineBid" 속성의 새 값을 설정.

```
bool  ShowLineBid(
   bool  show      // 속성 값
  \)
```

매개변수

표시/

[in] "ShowLineBid" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowLineAsk (Get Method)

"ShowLineAsk" 속성 값 가져오기.

```
bool  ShowLineAsk() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ShowLineAsk" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

ShowLineAsk (Set Method)

"ShowLineAsk" 속성의 새 값을 설정.

```
bool  ShowLineAsk(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowLineAsk" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowLastLine (Get Method)

"ShowLastLine" 속성 값 가져오기.

```
bool  ShowLastLine() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ShowLastLine" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

ShowLastLine (Set Method)

"ShowLastLine" 속성의 새 값 설정.

```
bool  ShowLastLine(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowLastLine" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowPeriodSep (Get Method)

"ShowPeriodSep" 속성 값(기간 구분자 표시)을 가져오기.

```
bool  ShowPeriodSep() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ShowPeriodSep" 속성 값. 할당된 차트가 없으면, false를 반환합니다.

ShowPeriodSep (Set Method)

"ShowPeriodSep" 속성의 새 값을 설정.

```
bool  ShowPeriodSep(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowPeriodSep" 속성 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowGrid (Get Method)

"ShowGrid" 속성의 값 가져오기.

```
bool  ShowGrid() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ShowGrid" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

ShowGrid (Set Method)

"ShowGrid" 속성 새 값 설정.

```
bool  ShowGrid(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowGrid" 속성 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowVolumes (Get Method)

"ShowVolumes" 속성 값을 가져오기.

```
bool  ShowVolumes() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ShowVolumes" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

ShowVolumes (Set Method)

"ShowVolumes" 속성의 새 값을 설정.

```
bool  ShowVolumes(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowVolumes" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowObjectDescr (Get Method)

"ShowObjectDescr" 속성 값(그래픽 개체에 대한 설명 표시) 가져오기.

```
bool  ShowObjectDescr() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ShowObjectDescr" 속성 값. 할당된 차트가 없으면 false를 반환합니다.

ShowObjectDescr (메서드 설정하기)

"ShowObjectDescr" 속성의 새 값을 설정.

```
bool  ShowObjectDescr(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowObjectDescr" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowDateScale

"ShowDateScale" 속성의 새 값을 설정.

```
bool  ShowDateScale(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowDateScale" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ShowPriceScale

"ShowPriceScale" 속성의 새 값을 설정.

```
bool  ShowPriceScale(
   bool  show      // 속성 값
  \)
```

매개변수

표시

[in] "ShowPriceScale" 속성의 새 값.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorBackground (Get Method)

"ColorBackground" 속성 값(차트의 배경 색상)을 가져오기.

```
color  ColorBackground() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorBackground" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorBackground (Set Method)

"ColorBackground" 속성의 새 값을 설정.

```
bool  ColorBackground(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 새로운 배경 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorForeground (Get Method)

"ColorForeground" 속성 값(차트의 축, 스케일 및 OHLC 문자열의 색상)을 가져오기.

```
color  ColorForeground() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorForeground" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorForeground (Set Method)

"ColorForeground" 속성(축, 스케일 및 OHLC 문자열)에 대한 새 값을 설정.

```
bool  ColorForeground(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 축, 스케일 및 OHLC 문자열의 새로운 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorGrid (Get Method)

"ColorGrid" 속성 값(그리드 색상)을 가져오기.

```
color  ColorGrid() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorGrid" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorGrid (Set Method)

"ColorGrid" 속성의 새 값을 설정.

```
bool  ColorGrid(
   color  new_color      // 색상
  \)
```

매개변수

new_color
  [in] 새 그리드 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorBarUp (Get Method)

"ColorBarUp" 속성 값(강세 막대의 색상, 그림자 및 캔들 본체 윤곽선)을 가져오기.

```
color  ColorBarUp() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorBarUp" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorBarUp (Set Method)

"ColorBarUp" 속성의 새 값을 설정.

```
bool  ColorBarUp(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 강세 막대의 새로운 색상, 그림자 및 캔들 본체 윤곽선.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorBarDown (Get Method)

"ColorBarDown" 속성 값(약세 막대의 색상, 그림자 및 캔들 본체 윤곽선)을 가져오기.

```
color  ColorBarDown() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorBarDown" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorBarDown (Set Method)

"ColorBarDown" 속성의 새 값을 설정.

```
bool  ColorBarDown(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 강세 막대의 새로운 색상, 그림자 및 캐들 본체 윤곽선.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorCandleBull (메서드 가져오기)

"ColorCandleBull" 속성 값(강세 캔들의 본체 색상)을 가져오기.

```
color  ColorCandleBull() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorCandleBull" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorCandleBull (Set Method)

"ColorCandleBull" 속성의 새 값을 설정.

```
bool  ColorCandleBull(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 강세 캔들 몸통의 새로운 컬러.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorCandleBear (메서드 가져오기)

"ColorCandleBear" 속성 값(약세 캔들의 본체 색상)을 가져오기.

```
color  ColorCandleBear() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorCandleBear" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorCandleBear (Set Method)

"ColorCandleBear" 속성의 새 값을 설정.

```
bool  ColorCandleBear(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 약세 캔들 본체의 새로운 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorChartLine (Get Method)

"ColorChartLine" 속성 값(선형 차트 및 도지 캔들의 색상)을 가져오기.

```
color  ColorChartLine() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorChartLine" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorChartLine (Set Method)

"ColorChartLine" 속성의 새 값을 설정.

```
bool  ColorChartLine(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 차트 선 및 도지의 캔들의 새로운 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorVolumes (Get Method)

"ColorVolumes" 속성 값(볼륨 및 열린 포지션의 수준에 대한 색상)을 가져오기.

```
color  ColorVolumes() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorVolumes" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorVolumes (Set Method)

"ColorVolumes" 속성의 새 값.

```
bool  ColorVolumes(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 볼륨의 새 색상과 열린 포지션의 수준.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorLineBid (Get Method)

"ColorLineBid" 속성 값(입찰 선 색상)을 가져오기.

```
color  ColorLineBid() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorLineBid" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorLineBid (Set Method)

"ColorLineBid" 속성의 새 값을 설정.

```
bool  ColorLineBid(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 입찰 선의 새 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorLineAsk (Get Method)

"ColorLineAsk" 속성 값(호가 선 색상)을 가져오기.

```
color  ColorLineAsk() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorLineAsk" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorLineAsk (Set Method)

"ColorLineAsk" 속성의 새 값을 설정.

```
bool  ColorLineAsk(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 호가 선의 새로운 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorLineLast (Get Method)

"ColorLineLast" 속성 값(마지막 거래 가격 선의 색상)을 가져오기.

```
color  ColorLineLast() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorLineLast" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorLineLast (Set Method)

"ColorLineLast" 속성의 새 값을 설정.

```
bool  ColorLineLast(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] 마지막 거래 가격 선의 새로운 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ColorStopLevels (Get Method)

"ColorStopLevels" 속성 값(SL 및 TP 레벨의 색상)을 가져오기.

```
color  ColorStopLevels() const
```

값 반환

클래스 인스턴스에 할당된 차트의 "ColorStopLevels" 속성 값. 할당된 차트가 없으면 CLR_NONE을 반환합니다.

ColorStopLevels (Set Method)

"ColorStopLevels" 속성의 새 값을 설정.

```
bool  ColorStopLevels(
   color  new_color      // 색상
  \)
```

매개변수

new_color

[in] Stop Loss 및 Take Profit price 수준의 새로운 색상.

값 반환

성공하면 true, 색상을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

VisibleBars

표시되는 차트 막대의 총 수 가져오기.

```
int  VisibleBars() const
```

값 반환

클래스 인스턴스에 할당된 차트의 총 표시 막대 수를 가져오기. 할당된 차트가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

WindowsTotal

차트 지표 하위 창을 포함하여 총 차트 창 수를 가져오기.

```
int  WindowsTotal() const
```

값 반환

차트 지표 하위 창을 포함하여 클래스 인스턴스에 할당된 총 창 수. 할당된 차트가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

WindowIsVisible

지정한 차트 하위 창의 가시성 플래그를 가져오기.

```
bool  WindowIsVisible(
   int  num      // 하위 창
   ) const
```

매개변수

num

[in] 하위 창 숫자 (0은 주 창을 의미).

값 반환

차트 인스턴스에 할당된 지정된 차트 하위 창의 가시성 플래그를 반환. 차트가 할당되지 않으면, false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

WindowHandle

차트의 창 핸들(HWND)을 가져오기.

```
int  WindowHandle() const
```

값 반환

차트 인스턴스에 할당된 차트의 창 핸들. 차트가 할당되지 않은 경우, INVALID_HANDLE을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

FirstVisibleBar

차트의 첫 번째 표시 막대 번호를 가져오기.

```
int  FirstVisibleBar() const
```

값 반환

클래스 인스턴스에 할당된 차트의 첫 번째 표시 막대 번호. 할당된 차트가 없으면 -1을 반환.

© 2000-2025, MetaQuotes Ltd.

WidthInBars

차트 너비(막대형) 가져오기.

```
int  WidthInBars() const
```

값 반환

클래스 인스턴스에 할당된 차트의 너비(차트 막대형). 할당된 차트가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

WidthInPixels

차트 너비(픽셀) 가져오기.

```
int  WidthInPixels() const
```

값 반환

차트 인스턴스에 할당된 차트 너비(픽셀). 할당된 차트가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

HeightInPixels

창 높이(픽셀) 가져오기.

```
int  HeightInPixels(
   int  num      // 하위 창
   ) const
```

매개변수

num

[in] 체크된 하위 창 번호(0은 주 창을 의미).

값 반환

클래스 인스턴스에 할당된 차트의 창 높이(픽셀). 할당된 차트가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

PriceMin

창 최소 가격 가져오기.

```
double  PriceMin(
   int  num      // 하위 창
   ) const
```

매개변수

num

[in] 하위 창 숫자 (0은 주 창을 의미).

값 반환

클래스 인스턴스에 할당된 차트의 최소 가격 값. 할당된 차트가 없으면, EMPTY_VALUE을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

PriceMax

창 최대 가격 가져오기.

```
double  PriceMax(
   int  num      // 하위 창
   ) const
```

매개변수

num

[in] 하위 창 숫자 (0은 주 창을 의미).

값 반환

클래스 인스턴스에 할당된 차트의 최대 가격 값. 할당된 차트가 없으면, EMPTY_VALUE을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Attach

현재 차트를 클래스 인스턴스에 할당하기

```
void  Attach()
```

Attach

지정한 차트를 클래스 인스턴스에 할당하기.

```
void  Attach(
   long  chart      // 차트 식별자
  \)
```

매개변수

차트

[in] 할당된 차트의 식별자.

© 2000-2025, MetaQuotes Ltd.

FirstChart

클라이언트 터미널의 첫 번째 차트를 클래스 인스턴스에 할당.

```
void  FirstChart()
```

© 2000-2025, MetaQuotes Ltd.

NextChart

클래스 인스턴스에 이미 할당된 차트 다음에 다음 차트를 할당합니다.

```
void  NextChart()
```

© 2000-2025, MetaQuotes Ltd.

Open

지정된 매개변수가 있는 차트를 열고 클래스 인스턴스에 할당.

```
long  Open(
   const string     symbol_name,     // 기호
   ENUM_TIMEFRAMES  timeframe        // 기간
  \)
```

매개변수

symbol_name

[in] 차트 기호. NULL은 현재 차트의 기호(전문가가 첨부됨)를 의미합니다.

시간 프레임

[in] 차트 시간 프레임 (ENUM_TIMEFRAMES 열거에서). 0은 현재 시간 프레임을 의미합니다.

값 반환

차트 식별자.

© 2000-2025, MetaQuotes Ltd.

Detach

차트를 클래스 인스턴스에서 분리.

```
void  Detach()
```

© 2000-2025, MetaQuotes Ltd.

Close

클래스 인스턴스에 할당된 차트를 닫기

```
void  Close()
```

© 2000-2025, MetaQuotes Ltd.

BringToTop

다른 차트 위에 차트를 표시.

```
bool  BringToTop() const
```

값 반환

성공하면 true, 오류에는 false.

© 2000-2025, MetaQuotes Ltd.

EventObjectCreate

그래픽 개체 생성 이벤트의 알림을 보내도록 플래그를 설정.

```
bool  EventObjectCreate(
   bool  flag      // 플래그
  \)
```

매개변수

Flags

[in] 그래픽 개체 생성 이벤트 알림을 보내는 플래그의 새 값.

값 반환

성공하면 true, 플래그를 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

EventObjectDelete

그래픽 개체 삭제 이벤트 알림을 보내도록 플래그를 설정.

```
bool  EventObjectDelete(
   bool  flag      // 플래그
  \)
```

매개변수

Flags

[in] 그래픽 개체 삭제 이벤트 알림을 보내는 플래그의 새 값.

값 반환

성공하면 true, 플래그를 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

IndicatorAdd

지정된 핸들을 가진 지표를 지정한 차트 창에 추가.

```
bool  IndicatorAdd(
   int   sub_win          // 하위창의 숫자
   int   handle           // 지표의 핸들
   );
```

매개변수

sub_win

[in] 차트 하위 창의 수. 0은 주 차트 창을 의미합니다. 존재하지 않는 창의 수를 지정하면 새 창이 생성됩니다.

핸들

[in] 지표의 핸들.

값 반환

이 함수는 성공할 경우 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 정보를 얻으려면, GetLastError() 함수를 호출합니다.

추가 참조

IndicatorDelete(), IndicatorsTotal(), IndicatorName().

© 2000-2025, MetaQuotes Ltd.

IndicatorDelete

지정된 차트 창에서 지정된 이름의 지표를 제거.

```
bool  IndicatorDelete(
   int            sub_win       // 하위 창의 숫자
   const string   name          // 지표의 단축명
   );
```

매개변수

sub_win

[in] 차트 하위 창의 숫자. 0은 주 차트 하위 창을 나타냅니다.

const name

[in] INDICATOR_SHORTNAME 속성에 IndicatorSetString() 함수와 함께 설정된 지표의 단축명. 지표의 단축명을 가져오려면 IndicatorName() 함수를 사용합니다.

값 반환

지표를 성공적으로 삭제할 경우 true를 반환. 그렇지 않으면 false가 반환됩니다. 오류 세부 정보를 가져오려면, GetLastError() 함수를 사용합니다.

참고

차트 하위 창에 동일한 단축 이름을 가진 두 개의 지표가 있으면 행의 첫 번째 지표가 삭제됩니다.

이 차트의 다른 지표가 삭제 중인 지표 값을 기반으로 하는 경우 해당 지표도 삭제됩니다.

iCustom() 및 IndicatorCreate() 함수를 사용하여 지표를 만들 때 지표 단축명과 지정한 Flename을 혼동하지 마십시오. 지표의 단축명이 명시적으로 설정되지 않은 경우, 지표의 소스 코드를 포함하는 Flename이 컴파일 중에 지정됩니다.

차트에서 지표를 삭제한다고 해서 계산 부분이 터미널 메모리에서 삭제되는 것은 아닙니다. 지표 핸들을 해제하려면 IndicatorRelease() 기능을 사용합니다.

지표의 짧은 이름이 올바르게 형성되어야 합니다. INDICATOR_SHORTNAME 속성에 IndicatorSetString() 함수를 사용하여 기록됩니다. IndicatorDelete() 함수에 의해 차트에서 삭제될 지표는 단축명으로 식별되므로 단축명에는 지표의 모든 입력 매개변수 값이 포함되어야 합니다.

추가 참조

IndicatorAdd(), IndicatorsTotal(), IndicatorName(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

IndicatorsTotal

지정된 차트 창에 적용된 모든 지표 수를 반환.

```
int  IndicatorsTotal(
   long   chart_id,  // 차트 식별자
   int    sub_win    // 하위 창의 숫자
   );
```

매개변수

chart_id

[in] 차트 식별자. 0은 주 차트를 나타냅니다.

sub_win

[in] 차트 하위 창의 숫자. 0은 주 차트 창을 나타냅니다.

값 반환

지정된 차트 창의 지표 수입니다. 오류 세부 정보를 가져오려면, GetLastError() 함수를 사용합니다.

참고

이 기능을 사용하면 차트에 첨부된 모든 지표를 검색할 수 있습니다. 차트의 모든 창 수는 CHART_WINDOWS_TOTAL 속성에서 GetInteger() 함수를 사용하여 얻을 수 있습니다.

추가 참조

IndicatorAdd(), IndicatorDelete(), IndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

IndicatorName

지정된 차트 창의 지표 목록에서 인덱스를 기준으로 지표의 단축명을 반환합니다.

```
string  IndicatorName(
   int   sub_win     // 하위 창의 숫자
   int   index       // 채팅 하위 창에 추가된 지표 목록 내 지표의 인덱스
   );
```

매개변수

sub_win

[in] 차트 하위 창의 숫자. 0은 주 차트 창을 나타냅니다.

인덱스

[in] 지표 목록에 있는 지표의 색인. 지표의 수는 0으로 시작합니다. 즉, 목록의 첫 번째 지표는 0 인덱스를 가집니다. 목록의 지표 수를 얻으려면 IndicatorsTotal() 함수를 사용합니다.

값 반환

INDICATOR_SHORTNAME 속성을 IndicatorSetString() 함수를 통해 설정된 지표 단축명. 오류 세부 정보를 가져오려면, GetLastError() 함수를 사용합니다.

참고

iCustom() 및 IndicatorCreate() 함수를 사용하여 지표를 만들 때 지표 단축명과 지정한 Flename을 혼동하지 마십시오. 지표의 단축명이 명시적으로 설정되지 않은 경우, 지표의 소스 코드를 포함하는 Flename이 컴파일 중에 지정됩니다.

지표의 짧은 이름이 올바르게 형성되어야 합니다. INDICATOR_SHORTNAME 속성에 IndicatorSetString() 함수를 사용하여 기록됩니다. IndicatorDelete() 함수에 의해 차트에서 삭제될 지표는 단축명으로 식별되므로 단축명에는 지표의 모든 입력 매개변수 값이 포함되어야 합니다.

추가 참조

IndicatorAdd(), IndicatorDelete, IndicatorsTotal, iCustom(), IndicatorCreate(), IndicatorSetString().

© 2000-2025, MetaQuotes Ltd.

Navigate

차트를 이동.

```
bool  Navigate(
   ENUM_CHART_POSITION  position,     // 포지션
   int                  shift=0       // 이동
  \)
```

매개변수

포지션

[in] 이동이 수행되는 차트 포지션 (ENUM_CHART_POSITION 열거에서).

shift=0

[in] 이동할 막대 수.

값 반환

성공하면 true, 차트를 이동할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

Symbol

차트 기호 이름 가져오기.

```
string  Symbol() const
```

값 반환

차트의 기호 이름, 클래스 인스턴스에 할당됨. 차트가 할당되지 않은 경우 ""를 반환.

© 2000-2025, MetaQuotes Ltd.

Period

차트의 기간을 가져오기.

```
ENUM_TIMEFRAMES  Period() const
```

값 반환

차트 기간(ENUM_TIMEFRAMES에서)이 클래스 인스턴스에 할당되었습니다. 할당된 차트가 없으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Redraw

클래스 인스턴스에 할당된 차트를 다시 그리기.

```
void  Redraw()
```

© 2000-2025, MetaQuotes Ltd.

GetInteger

함수는 해당 차트 속성의 값을 반환합니다. 차트 속성은 정수 유형이어야 합니다. 함수에는 두 가지 유형이 있습니다.

1. 속성 값을 즉시 반환.

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // 속성 식별자
   int                          sub_window=0      // 하위 창 숫자
   ) const
```

2. 성공하면 속성 값을 지정된 정수로 유형의 변수에 넣고 참조에서 마지막 매개변수로 전달합니다.

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,        // 속성 식별자
   int                          sub_window,     // 하위 창 숫자
   long&                        value           // 변수에 연결
   ) const
```

매개변수

prop_id

[in] 속성 식별자 (ENUM_CHART_PROPERTY_INTEGER 열거).

sub_window

[in] 차트 하위 창 숫자.

값

[in] 요청한 속성의 값을 수신하는 변수에 대한 링크.

값 반환

클래스 인스턴스에 할당된 차트의 속성의 값. 할당된 차트가 없으면 -1을 반환합니다.

두 번째 변형에서 함수는 true를 반환합니다. 이 속성이 유지되고 값이 값 변수에 배치되면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용을 보려면, GetLastError()에 문의합니다.

© 2000-2025, MetaQuotes Ltd.

SetInteger

정수 유형의 속성에 대한 새 값을 설정.

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // 속성 식별자
   long                         value         // 값
  \)
```

매개변수

prop_id

[in] 차트 속성 식별자 (ENUM_CHART_PROPERTY_INTEGER 열거에서).

값

[in] 속성의 새 값.

값 반환

성공하면 true, 정수 속성을 변경하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

GetDouble

함수는 해당 차트 속성의 값을 반환합니다. 개체 속성은 double 형식이어야 합니다. 함수에는 두 가지 유형이 있습니다.

1. 속성 값을 즉시 반환.

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // 속성 식별자
   int                         sub_window=0     // 하위 창 숫자
   ) const
```

2. 성공하면 속성 값을 지정된 double 유형의 변수에 넣고 참조에서 마지막 매개변수로 전달합니다.

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,        // 속성 식별자
   int                         sub_window,     // 하위 창 숫자
   double&                     value           // 변수에 연결
   ) const
```

매개변수

prop_id

[in] 차트 속성 식별자 (ENUM_CHART_PROPERTY_DOUBLE 열거에서).

sub_window

[in] 차트 하위 창 숫자.

값

[in] 요청한 속성의 값을 받은 double 유형의 변수.

값 반환

클래스 인스턴스에 할당된 차트의 속성 값. 할당된 차트가 없으면 EMPTY_VALUE를 반환합니다.

두 번째 변형 함수의 경우 속성 값이 수신되면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용을 보려면, GetLastError()를 호출합니다.

© 2000-2025, MetaQuotes Ltd.

SetDouble

double 유형의 차트 속성에 대한 새 값을 설정.

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // 속성 식별자
   double                      value         // 새 값
  \)
```

매개변수

prop_id

[in] 차트 속성 식별자 (ENUM_CHART_PROPERTY_DOUBLE 열거에서).

값

[in] 속성의 새 값.

값 반환

성공하면 true, double 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

GetString

함수는 해당 차트 속성의 값을 반환합니다. 차트 속성은 문자열 유형이어야 합니다. 함수에는 두 가지 유형이 있습니다.

1. 속성 값을 즉시 반환.

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // 속성 식별자
   ) const
```

2. 성공하면 속성 값을 지정된 문자열 유형의 변수에 넣고 참조에서 마지막 매개 변수로 전달합니다.

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // 속성 식별자
   string&                     value        // 변수에 연결
   ) const
```

매개변수

prop_id

[in] 차트 속성 식별자 (ENUM_CHART_PROPERTY_STRING 열거에서).

값

[in] 요청한 속성의 값을 수신하는 변수에 대한 링크.

값 반환

클래스 인스턴스에 할당된 차트 속성의 값. 차트가 할당되지 않은 경우 ""를 반환.

두 번째 변형에서 함수는 true를 반환합니다. 이 속성이 유지되고 값이 값 변수에 배치되면 true를 반환하고 그렇지 않으면 false를 반환합니다. 오류에 대한 자세한 내용을 보려면, GetLastError()에 문의합니다.

© 2000-2025, MetaQuotes Ltd.

SetString

문자열 유형의 차트 속성에 대한 새 값을 설정.

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // 속성 식별자
   string                      value         // 값
  \)
```

매개변수

prop_id

[in] 차트 속성 식별자 (ENUM_CHART_PROPERTY_STRING 열거에서).

값

[in] 속성의 새 값.

값 반환

성공하면 true, 문자열 속성을 변경하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

클래스 인스턴스에 할당된 차트의 기호 및 기간을 변경.

```
bool  SetSymbolPeriod(
   const string      symbol_name,      // 기호
   ENUM_TIMEFRAMES   timeframe         // 기간
  \)
```

매개변수

symbol_name

[in] 새 차트 기호. NULL은 현재 차트의 기호(전문가가 첨부됨)를 의미합니다.

시간 프레임

[in] 새 차트 시간 프레임 (ENUM_TIMEFRAMES 열거에서). 0은 현재 차트 시간 프레임을 의미합니다.

값 반환

성공하면 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

ApplyTemplate

차트에 지정된 템플릿 적용하기.

```
bool  ApplyTemplate(
   const string  filename      // 템플릿
  \)
```

매개변수

Flename

[in] 템플릿의 파일 이름.

값 반환

성공하면 true, 템플릿을 적용하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

ScreenShot

지정한 차트의 스크린샷을 현재 상태로 .gif 형식으로 생성.

```
bool  ScreenShot(
   string           filename,                  // 파일 이름
   int              width,                     // 너비
   int              height,                    // 높이
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // 정렬 유형
   ) const
```

매개변수

Flename

[in] 스크린샷 파일 이름.

width

[in] 스크린샷 너비(픽셀).

height

[in] 스크린샷 높이(픽셀).

align_mode=ALIGN_RIGHT

[in] 스크린 샷이 좁은 경우 정렬 모드.

값 반환

성공하면 true, 오류에는 false.

© 2000-2025, MetaQuotes Ltd.

WindowOnDropped

개체(전문가 또는 스크립트) 드롭 포인트에 해당하는 차트 하위 창 번호 가져오기.

```
int  WindowOnDropped() const
```

값 반환

개체 드롭 포인트의 하위 창 번호를 차트화 0은 주 창을 의미합니다.

© 2000-2025, MetaQuotes Ltd.

PriceOnDropped

개체(전문가 또는 스크립트)의 하락점에 해당하는 가격 좌표 가져오기.

```
double  PriceOnDropped() const
```

값 반환

개체 하락점의 가격 좌표.

© 2000-2025, MetaQuotes Ltd.

TimeOnDropped

개체(전문가 또는 스크립트) 드롭 포인트에 (expert or script) drop point.

```
datetime  TimeOnDropped() const
```

값 반환

개체 드롭 포인트의 시간 좌표.

© 2000-2025, MetaQuotes Ltd.

XOnDropped

개체(전문가 또는 스크립트) 드롭 포인트에 해당하는 X 좌표.

```
int  XOnDropped() const
```

값 반환

개체 드롭 포인트 X 좌표.

© 2000-2025, MetaQuotes Ltd.

YOnDropped

개체(전문가 또는 스크립트) 드롭 포인트에 해당하는 Y 좌표 가져오기.

```
int  YOnDropped() const
```

값 반환

개체 드롭 포인트 Y 좌표.

© 2000-2025, MetaQuotes Ltd.

Save

개체 매개변수를 파일에 저장.

```
virtual bool  Save(
   int  file_handle      // 파일 핸들
  \)
```

매개변수

file_handle

[in] FileOpen(...) 함수로 이미 열려 있는 binary 파일의 핸들.

값 반환

성공하면 true, 오류에는 false.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 개체 매개변수 불러오기.

```
virtual bool  Load(
   int  file_handle      // 파일 핸들
  \)
```

매개변수

file_handle

[in] FileOpen(...) 함수로 이미 열려 있는 binary 파일의 핸들.

값 반환

성공하면 true, 오류에는 false.

© 2000-2025, MetaQuotes Ltd.

Type

유형 식별자 반환.

```
virtual int  Type() const
```

값 반환

유형 식별자(CChart 용 0x1111).

© 2000-2025, MetaQuotes Ltd.

Graphics

그래픽 라이브러리에는 사용자 지정 차트를 빠르게 표시하기 위한 클래스 및 글로벌 함수가 포함되어 있습니다. 라이브러리는 축, 곡서을 작성할 수 있는 편리한 기성 솔루션뿐만 아니라 사용자 지정 차트의 변화하는 공통 속성에 빠르게 액세스할 수 있는 방법을 제공합니다.

라이브러리로 작업을 시작하려면 다음 아티클을 읽기만 하면 됩니다: Visualize this! R 언어의 'plot'과 유사한 MQL5 그래픽 라이브러리.

그래픽 라이브러리는 터미널 작업 디렉토리의 Include₩Graphics 폴더에 배치됩니다.

클래스	설명
CAxis	좌표 축에 대한 작업을 위한 클래스
ColorGenerator	기본 색상 구성표를 지정하는 클래스
CCurve	곡선 작업을 위한 클래스
CGraphic	사용자 지정 차트를 만들기 위한 기본 클래스

© 2000-2025, MetaQuotes Ltd.

GraphPlot

빠른 곡선 플로팅 기능.

Y 좌표를 사용하여 단일 곡선을 플로팅하기 위한 버전.

```
string  GraphPlot(
   const double      &y[],                 // Y 좌표
   ENUM_CURVE_TYPE   type=CURVE_POINTS     // 곡선 유형
  \)
```

참고

Y 배열 지수는 곡선의 X 좌표로 사용됩니다.

X 및 Y 좌표를 사용하여 단일 곡선을 플로팅하기 위한 버전

```
string  GraphPlot(
   const double      &x[],                 // X 좌표
   const double      &y[],                 // Y 좌표
   ENUM_CURVE_TYPE   type=CURVE_POINTS     // 곡선 유형
  \)
```

X 및 Y 좌표를 사용하여 두 곡선을 플로팅하기 위한 버전

```
string  GraphPlot(
   const double      &x1[],                // X 좌표
   const double      &y1[],                // Y 좌표
   const double      &x2[],                // X 좌표
   const double      &y2[],                // Y 좌표
   ENUM_CURVE_TYPE   type=CURVE_POINTS     // 곡선 유형
  \)
```

X 및 Y 좌표를 사용하여 세 곡선을 플로팅하기 위한 버전

```
string  GraphPlot(
   const double      &x1[],                // X 좌표
   const double      &y1[],                // Y 좌표
   const double      &x2[],                // X 좌표
   const double      &y2[],                // Y 좌표
   const double      &x3[],                // X 좌표
   const double      &y3[],                // Y 좌표
   ENUM_CURVE_TYPE   type=CURVE_POINTS     // 곡선 유형
  \)
```

© 2000-2025, MetaQuotes Ltd.

CPoint2D 점 좌표를 사용하여 곡선을 플로팅하는 버전

```
string  GraphPlot(
   const CPoint2D   &points[],            // 곡선 좌표
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 곡선 유형
  \)
```

CPoint2D 점 좌표를 사용하여 두 곡선을 플로팅하기 위한 버전

```
string  GraphPlot(
   const CPoint2D   &points1[],           // 곡선 좌표
   const CPoint2D   &points2[],           // 곡선 좌표
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 곡선 유형
  \)
```

CPoint2D 점 좌표를 사용하여 세 곡선을 플로팅하기 위한 버전

```
string  GraphPlot(
   const CPoint2D   &points1[],           // 곡선 좌표
   const CPoint2D   &points2[],           // 곡선 좌표
   const CPoint2D   &points3[],           // 곡선 좌표
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 곡선 유형
  \)
```

CurveFunction에 대한 포인터를 사용하여 곡선을 플로팅하기 위한 버전

```
string  GraphPlot(
   CurveFunction    function,             // 함수에 대한 포인터
   const double     from,                 // 인수의 초기값
   const double     to,                   // 인수의 최종값
   const double     step,                 // 인수 증가
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 곡선 유형
  \)
```

CurveFunction 함수에 대한 포인터를 사용하여 두 곡선을 플로팅하기 위한 버전

```
string  GraphPlot(
   CurveFunction    function1,            // 함수에 대한 포인터
   CurveFunction    function2,            // 함수에 대한 포인터
   const double     from,                 // 인수의 초기값
   const double     to,                   // 인수의 최종값
   const double     step,                 // 인수 증가
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // 곡선 유형
  \)
```

© 2000-2025, MetaQuotes Ltd.

CurveFunction 함수에 대한 포인터를 사용하여 세 곡선을 플로팅하기 위한 버전

```
string  GraphPlot(
   CurveFunction    function1,             // 함수에 대한 포인터
   CurveFunction    function2,             // 함수에 대한 포인터
   CurveFunction    function3,             // 기능에 대한 포인터
   const double     from,                  // 인수의 초기값
   const double     to,                    // 인수의 최종값
   const double     step,                  // 인수 증가
   ENUM_CURVE_TYPE  type=CURVE_POINTS      // 곡선 유형
  \)
```

매개변수

&x[]
  [in] X 좌표.

&y[]
  [in] Y 좌표.

&x1[]
  [in] 첫 번째 곡선의 X 좌표.

&y1[]
  [in] 첫 번째 곡선의 Y 좌표.

&x2[]
  [in] 두 번째 곡선의 X 좌표.

&y2[]
  [in] 두 번째 곡선의 Y 좌표.

&x3[]
  [in] 세 번째 곡선의 X 좌표.

&y3[]
  [in] 세 번째 곡선의 Y 좌표.

&points[]
  [in] 곡선 점의 좌표.

&points1[]
  [in] 첫 번째 곡선 점의 좌표.

&points2[]
  [in] 두 번째 곡선 점의 좌표.

&points3[]
  [in] 세 번째 곡선 점의 좌표.

© 2000-2025, MetaQuotes Ltd.

함수

[in] CurveFunction 함수에 대한 포인터.

function1

[in] 첫 번째 함수에 대한 포인터.

function2

[in] 두 번째 함수에 대한 포인터.

function3

[in] 세 번째 함수에 대한 포인터.

from

[in] 첫 번째 X 좌표에 해당.

to

[in] 마지막 X 좌표에 해당.

step

[in] X 좌표를 계산하기 위한 매개변수.

type=CURVE_POINTS

[in] 곡선 유형.

값 반환

그래픽 리소스의 이름.

© 2000-2025, MetaQuotes Ltd.

CAxis

CAxis는 좌표 축으로 작업하기 위한 보조 그래픽 라이브러리 클래스입니다.

설명

CAxis 클래스는 좌표 축이 다양한 매개변수를 수신하고 저장합니다. 클래스는 좌표 축의 크기를 동적으로 자동 조정할 수 있는 기능을 구현합니다.

선언

```
class CAxis
```

제목

```
#include <Graphics\Axis.mqh>
```

클래스 메서드

메서드	설명
AutoScale	자동 스케일링 플래그 가져오기/설정하기
Min	최소 축 값을 가져오기/설정하기
Max	최대 축 값을 가져오기/설정하기
Step	축별 단계 값 가져오기
Name	축 이름 가져오기/설정하기
Color	축 색상 가져오기/설정하기
ValuesSize	축 숫자의 크기 가져오기/설정하기
ValuesWidth	축 숫자의 표시된 최대 길이 가져오기/설정하기
ValuesFormat	축 숫자 형식 가져오기/설정하기
ValuesDateTimeMode	날짜를 문자열로 변환하는 형식으로 가져오기.
ValuesFunctionFormat	축에 값을 표시하는 형식을 정의하는 함수에 대한 포인터 가져오기.
ValuesFunctionFormatCBData	축 값 변환에 대한 추가 데이터를 포함할 수 있는 개체에 대한 포인터 가져오기.
NameSize	축 이름의 글꼴 크기 가져오기/설정하기
ZeroLever	"0 레버" 값 가져오기/설정하기
DefaultStep	축별 초기 단계 값 가져오기/설정하기
MaxLabels	축의 최대 숫자 양을 가져오기/설정하기
MinGrace	축 최소값에 대한 "tolerance" 값 가져오기/설정하기

© 2000-2025, MetaQuotes Ltd.

メソッド	説明
MaxGrace	축 최대값에 대한 "tolerance" 값 가져오기
SelectAxisScale	축 자동 스케일.

© 2000-2025, MetaQuotes Ltd.

AutoScale (Get method)

자동 스케일 필요성을 정의하는 플래그를 반환.

```
bool  AutoScale()
```

값 반환

플래그 값.

참고

true — 자동 스케일링 수행.

false — 자동 스케일링 수행하지 않음.

AutoScale (Set method)

자동 스케일 필요성을 정의하는 플래그를 설정하기.

```
void  AutoScale(
   const bool  auto      // 플래그 값
  \)
```

매개변수

auto

[in]

참고

true — 자동 스케일링 수행.

false — 자동 스케일링 수행하지 않음.

© 2000-2025, MetaQuotes Ltd.

Min (Get method)

최소 축 값을 반환.

```
double  Min()
```

값 반환

최소 축 값.

Min (Set method)

최소 축 값을 설정하기.

```
void  Min(
   const double  min      // 최소 값
  \)
```

매개변수

min

[in] 최소 값.

© 2000-2025, MetaQuotes Ltd.

Max (Get method)

최대 축 값을 반환.

```
double  Max()
```

값 반환

최대 축 값.

Max (Set method)

최대 축 값을 설정.

```
void  Max(
   const double  max      // 최대 값
  \)
```

매개변수

max

[in] 최대 축 값.

© 2000-2025, MetaQuotes Ltd.

Step (Get method)

단계 값을 축별로 반환.

```
double  Step()
```

값 반환

단계 값.

© 2000-2025, MetaQuotes Ltd.

Name (Get method)

축 이름 반환.

```
string  Name()
```

값 반환

축 이름.

Name (Set method)

축 이름 설정.

```
void  Name(
   const string  name      // 축 이름
  \)
```

매개변수

name

[in] 축 이름.

© 2000-2025, MetaQuotes Ltd.

Color (Get method)

축 색상 반환.

```
color  Color()
```

값 반환

축 색상.

Color (Set method)

축 색상 설정.

```
void  Color(
   const color  clr      // 축 색상
  \)
```

매개변수

clr

[in] 축 색상.

© 2000-2025, MetaQuotes Ltd.

ValuesSize (Get method)

축 번호의 크기를 반환.

```
int  ValuesSize()
```

값 반환

축 숫자의 크기.

ValuesSize (Set method)

축 숫자의 크기를 설정.

```
void  ValuesSize(
   const int  size      // 축 숫자의 크기
  \)
```

매개변수

Size

[in] 축 숫자 크기

© 2000-2025, MetaQuotes Ltd.

ValuesWidth (Get method)

축 숫자를 표시하기 위해 허용되는 최대 길이(픽셀)를 반환.

```
int  ValuesWidth()
```

값 반환

축 숫자 길이(픽셀).

참고

지정된 숫자의 픽셀 길이가 허용된 최대 표시 길이를 초과하면 잘리고 점으로 끝납니다.

ValuesWidth (Set method)

축 숫자를 표시할 수 있는 최대 허용 길이를 픽셀 단위로 설정.

```
void  ValuesWidth(
   const int  width      // 허용된 최대 길이(픽셀)
  \)
```

매개변수

width

[in] 축 숫자의 허용되는 최대 길이.

참고

지정된 숫자의 픽셀 길이가 허용된 최대 표시 길이를 초과하면 잘리고 점으로 끝납니다.

© 2000-2025, MetaQuotes Ltd.

ValuesFormat (Get method)

축 숫자 형식을 반환.

```
string  ValuesFormat()
```

값 반환

숫자 형식.

ValuesFormat (Set method)

축 숫자의 형식을 설정.

```
void  ValuesFormat(
   const string  format      // 축 숫자 형식
  \)
```

매개변수

형식

[in] 축 숫자 형식.

© 2000-2025, MetaQuotes Ltd.

ValuesDateTimeMode (Get method)

날짜를 문자열로 변환하는 형식으로 가져오기.

```
int  ValuesDateTimeMode()
```

값 반환

날짜를 문자열로 변환하는 형식.

ValuesDateTimeMode (Set method)

날짜를 문자열로 변환하는 형식으로 설정.

```
void  ValuesDateTimeMode(
   const int  mode      // 날짜를 문자열로 변환하는 형식
  \)
```

매개변수

mode

[in] 변환 형식.

참고

날짜를 문자열로 변환하는 형식에 대한 자세한 내용은 TimeToString() 함수 설명을 참조하십시오.

© 2000-2025, MetaQuotes Ltd.

ValuesFunctionFormat (Get method)

축에 값을 표시하는 형식을 정의하는 함수에 대한 포인터 가져오기.

```
DoubleToStringFunction  ValuesFunctionFormat()
```

값 반환

축에 값을 표시하는 형식을 정의하는 함수에 대한 포인터.

ValuesFunctionFormat (Set method)

축에 값을 표시하는 형식을 정의하는 함수로 포인터를 설정.

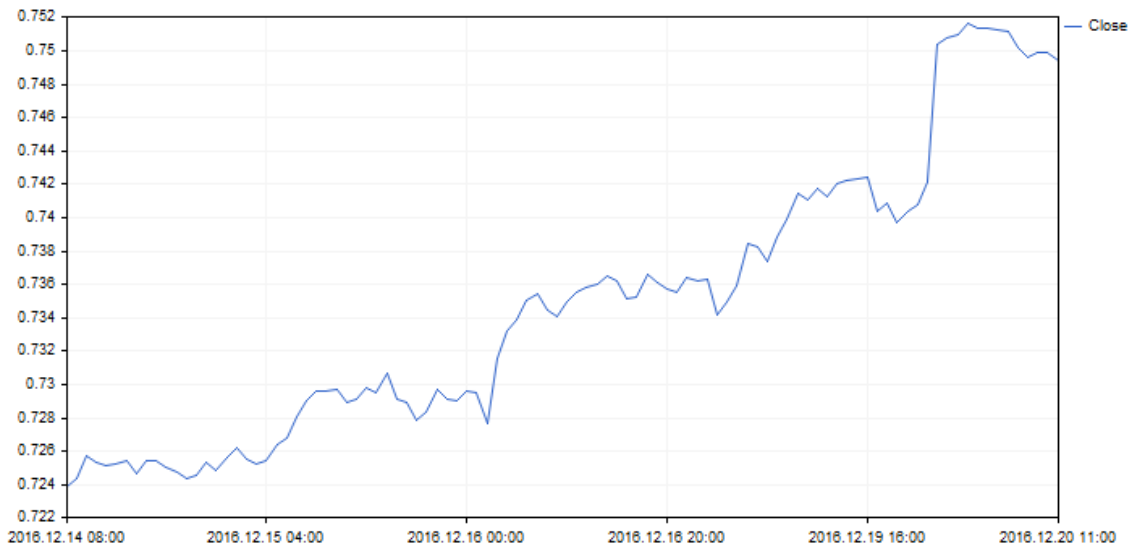
```
void  ValuesFunctionFormat(
   DoubleToStringFunction  func      // 숫자 값을 문자열로 변환하는 함수
  \)
```

매개변수

func

[in] 숫자 값을 문자열로 변환하기 위한 사용자 지정 함수.

예시:



다음 코드를 사용하여 X 축 값을 표시하는 형식이 변경되었습니다:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              DateAxisGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//--- 저장소 값의 배열
double arrX[];
double arrY[];
//+------------------------------------------------------------------+
//| X-축에 값을 만드는 사용자 지정 함수                              |
//+------------------------------------------------------------------+
string TimeFormat(double x,void *cbdata)
  {
   return(TimeToString((datetime)arrX[ArraySize(arrX)-(int)x-1]));
  }
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlRates rates[];
   CopyRates(Symbol(),Period(),0,100,rates);
   ArraySetAsSeries(rates,true);
   int size=ArraySize(rates);
   ArrayResize(arrX,size);
   ArrayResize(arrY,size);
   for(int i=0; i<size;++i)
     {
      arrX[i]=(double)rates[i].time;
      arrY[i]=rates[i].close;
     }
//--- 그래픽 생성
   CGraphic graphic;
   if(!graphic.Create(0,"DateAxisGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"DateAxisGraphic");
     }
//--- 곡선 생성
   CCurve *curve=graphic.CurveAdd(arrY,CURVE_LINES);
//--- X-축 가져오기
   CAxis *xAxis=graphic.XAxis();
//--- X-축 속성 설정하기
   xAxis.AutoScale(false);
   xAxis.Type(AXIS_TYPE_CUSTOM);
   xAxis.ValuesFunctionFormat(TimeFormat);
   xAxis.DefaultStep(20.0);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

ValuesFunctionFormatCBData (Get method)

축 값 변환에 대한 추가 데이터를 포함할 수 있는 개체에 대한 포인터 가져오기.

```
void*  ValuesFunctionFormatCBData()
```

값 반환

축 값 변환에 대한 추가 데이터를 포함할 수 있는 개체에 대한 포인터.

ValuesFunctionFormatCBData (Set method)

축 값 변환에 대한 추가 데이터가 포함될 수 있는 클래스 개체로 포인터를 설정.

```
void  ValuesFunctionFormatCBData(
   void*  cbdata      // 클래스 개체에 대한 포인터
  \)
```

매개변수

cbdata

[in] 축 값 변환에 대한 추가 데이터가 포함된 클래스 개체에 대한 포인터

© 2000-2025, MetaQuotes Ltd.

NameSize (Get method)

축 이름의 글꼴 크기 반환.

```
int  NameSize()
```

값 반환

축 이름의 글꼴 크기.

NameSize (Set method)

축 이름의 글꼴 크기 설정.

```
void  NameSize(
   const int  size      // 축 이름의 글꼴 크기
  \)
```

매개변수

Size

[in] 축 이름의 글꼴 크기.

© 2000-2025, MetaQuotes Ltd.

ZeroLever (Get method)

"제로 레버" 값을 반환.

```
double  ZeroLever()
```

값 반환

"제로 레버".

참고

이 값은 축 척도 범위가 0 값을 포함하도록 확장되어야 하는 시기를 정의하는 데 사용됩니다.

ZeroLever (Set method)

"제로 레버" 값 설정.

```
void  ZeroLever(
   const double  value      // "제로 레버" 값
  \)
```

매개변수

값

[in] "제로 레버" 값.

참고

이 값은 축 척도 범위가 0 값을 포함하도록 확장되어야 하는 시기를 정의하는 데 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

DefaultStep (Get method)

축별로 초기 단계 값을 반환

```
double  DefaultStep()
```

값 반환

축별 단계.

DefaultStep (Set method)

축별로 초기 단계 값을 설정하기

```
void  DefaultStep(
   const double  value      // 축별 단계
  \)
```

매개변수

값

[in] 축별 초기 단계.

© 2000-2025, MetaQuotes Ltd.

MaxLabels (Get method)

축에 표시되는 허용 최대 숫자 양을 반환.

```
double  MaxLabels()
```

값 반환

축의 최대 숫자 양.

MaxLabels (Set method)

축에 표시되는 최대 허용 숫자의 양을 설정.

```
void  MaxLabels(
   const double  value      // 최대 숫자
  \)
```

매개변수

값

[in] 축에 표시된 최대 허용 숫자의 양

© 2000-2025, MetaQuotes Ltd.

MinGrace (Get method)

축 최소값에 적용되는 "tolerance"를 반환.

```
double  MinGrace()
```

값 반환

축 최소값에 대한 "Tolerance" 값.

참고

이 값은 전체 축 길이의 일부로 표시. 여를 들어 축 값이 4.0～16.0 사이에 있다고 가정하면 축의 길이는 12.0입니다. MinGrace가 0.1이면 최소 값에서 축 길이의 10%(또는 1.2)를 뺍니다. 결과적으로 축은 2.8～16.0의 간격을 포함합니다.

MinGrace (Set method)

축 최소값에 적용되는 "tolerance" 를 설정.

```
void  MinGrace(
   const double  value      // "tolerance" 값
  \)
```

매개변수

값

[in] 축 최소값에 "Tolerance"가 적용.

참고

이 값은 전체 축 길이의 일부로 표시. 여를 들어 축 값이 4.0～16.0 사이에 있다고 가정하면 축의 길이는 12.0입니다. MinGrace가 0.1이면 최소 값에서 축 길이의 10%(또는 1.2)를 뺍니다. 결과적으로 축은 2.8～16.0의 간격을 포함합니다.

© 2000-2025, MetaQuotes Ltd.

MaxGrace (Get method)

축 최대값에 적용되는 "tolerance"를 반환.

```
double  MaxGrace()
```

값 반환

최대 축에 대한 "Tolerance" 값.

참고

이 값은 전체 축 길이의 일부로 표시. 여를 들어 축 값이 4.0～16.0 사이에 있다고 가정하면 축의 길이는 12.0입니다. MaxGrace가 0.1이면 축 길이의 10%(또는 1.2)가 최대값에 추가됩니다. 결과적으로 축은 4.0～17.2의 간격을 포함합니다.

MaxGrace (Set method)

축 최대값에 적용되는 "tolerance"를 설정하기.

```
void  MaxGrace(
   const double  value      // "tolerance" 값
  \)
```

매개변수

값

[in] 축 최대값에 적용된 "Tolerance" 값.

참고

이 값은 전체 축 길이의 일부로 표시. 여를 들어 축 값이 4.0～16.0 사이에 있다고 가정하면 축의 길이는 12.0입니다. MinGrace가 0.1이면 최소 값에서 축 길이의 10%(또는 1.2)를 뺍니다. 결과적으로 축은 2.8～16.0의 간격을 포함합니다.

© 2000-2025, MetaQuotes Ltd.

SelectAxisScale

축 자동 스케일.

```
void  SelectAxisScale()
```

© 2000-2025, MetaQuotes Ltd.

CColorGenerator

CColorGenerator 클래스는 색상 팔레트 작업을 위한 보조 그래픽 라이브러리 클래스입니다.

설명

CColorGenerator 클래스에는 기본적으로 곡선에 사용되는 초기 색상 팔레트가 포함되어 있습니다(색상이 사용자에 의해 지정되지 않은 경우).

초기 팔레트의 모든 색상이 이미 사용된 경우 새 색상이 자동으로 생성되고 팔레트가 다시 채워집니다.

선언

```
class CColorGenerator
```

제목

```
#include <Graphics\ColorGenerator.mqh>
```

클래스 메서드

메서드	설명
Next	팔레트에서 다음 색상을 반환
Reset	제너레이터를 재설정

© 2000-2025, MetaQuotes Ltd.

Next

팔레트에서 다음 색상을 반환.

```
uint  Next()
```

값 반환

색상.

참고

팔레트의 모든 색상이 이미 전달된 경우 팔레트의 이전 색상을 대체하기 위해 새 색상이 자동으로 생성됩니다.

© 2000-2025, MetaQuotes Ltd.

Reset

제너레이터 재설정.

```
void  Reset()
```

© 2000-2025, MetaQuotes Ltd.

CCurve

CCurve 클래스는 차트에 생성된 곡선의 속성으로 작동합니다.

설명

CCurve 클래스는 CGraphic 클래스로 작업할 때 곡선의 좌표 및 다양항 속성을 설정, 설치 및 수신합니다.

점, 선 및 히스토그램의 세 가지 곡선 표시 모드가 있습니다. 클래스의 각 플로팅 모드에 대해 별도의 매개변수가 구현됩니다.

선언

```
class CCurve : public CObject
```

제목

```
#include <Graphics\Curve.mqh>
```

상속 계층

CObject
  CCurve

클래스 메서드

메서드	설명
Type	곡선 유형 가져오기
Name	곡선 이름 가져오기
Color	곡선 색상 가져오기
XMax	X 함수의 최대값을 가져오기
XMin	X 함수 최소값을 가져오기
YMax	Y 함수 최대값을 가져오기
YMin	Y 함수 최소값을 가져오기
Size	곡선을 정의하는 점 수 가져오기
PointsSize	곡선을 정의하는 점의 선형 크기 가져오기
PointsFill	곡선을 정의하는 점을 채우기 위한 플래그 가져오기/설정하기
PointsColor	점 채우기 색상 가져오기/설정하기
GetX	모든 곡선 점의 X 값을 배열로 가져오기
GetY	모드 곡선 점의 Y 값을 배열로 가져오기

© 2000-2025, MetaQuotes Ltd.

メソッド	説明
LinesStyle	선을 사용하여 곡선을 표시할 때 선 스타일 가져오기/설정하기
LinesIsSmooth	선을 사용하여 그릴 때 평활 플래그 가져오기/설정하기
LinesSmoothTension	선을 사용하여 그릴 때 곡선 평활화 매개변수 가져오기/설정하기
LinesSmoothStep	선으로 표시할 때 평활을 위한 근사 선의 길이 가져오기/설정하기
LinesWidth	선을 사용하여 곡선을 표시할 때 선 너비 가져오기/설정하기
HistogramWidth	히스토그램을 사용하여 표시할 때 열 너비 가져오기/설정하기
CustomPlotCBData	사용자 지정 곡선 플로팅 모드에서 사용할 개체에 대한 포인터를 가져오기/설정하기.
CustomPlotFunction	사용자 지정 곡선 플로팅 모드를 구현하는 함수에 대한 포인터를 가져오기/설정하기.
PointsType	점선 곡선을 플로팅할 때 사요되는 점 유형을 가리키는 플래그를 가져오거나 설정.
StepsDimension	단계 유형 곡선 렌더링에 사용되는 치수를 나타내는 값을 가져오기/설정하기.
TrendLineCoefficients	배열에 쓰기 위한 추세선 비율을 가저오기/설정하기.
TrendLineColor	곡선에 대한 추세선 색상을 가져오기/설정하기.
TrendLineVisible	추세선 가시성 플래그를 가져오기/설정하기.
Update	곡선 좌표를 업데이트.
Visible	차트에 함수가 표시되는지 여ㅣ부를 정의하는 플래그를 가져오기/설정하기.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Compare

© 2000-2025, MetaQuotes Ltd.

Type

곡선 유형 반환.

```
ENUM_CURVE_TYPE   Type()
```

값 반환

곡선 유형.

© 2000-2025, MetaQuotes Ltd.

Name

곡선 이름 반환.

```
string  Name()
```

값 반환

곡선 이름.

© 2000-2025, MetaQuotes Ltd.

Color

곡선 색상을 반환.

```
uint  Color()
```

값 반환

곡선 색상.

© 2000-2025, MetaQuotes Ltd.

XMax

X 함수의 최대 값을 반환 (실수만 해당).

```
double  XMax()
```

값 반환

모든 함수 인수 중 최대의 실수.

© 2000-2025, MetaQuotes Ltd.

XMin

X 함수의 최소값을 반환 (실수만 해당).

```
double  XMin()
```

값 반환

모든 함수인수 중 최소의 실수.

© 2000-2025, MetaQuotes Ltd.

YMax

Y 함수의 최대값을 반환 (실수만 해당).

```
double  YMax()
```

값 반환

Y 함수의 최대값 (실수만 해당).

© 2000-2025, MetaQuotes Ltd.

YMin

Y 함수의 최소값을 반환 (실수만 해당).

```
double  YMin()
```

값 반환

Y 함수의 최소 값 (실수만 해당).

© 2000-2025, MetaQuotes Ltd.

Size

곡선을 정의하는 점 수를 반환.

```
int  Size()
```

값 반환

곡선을 정의하는 점 수.

© 2000-2025, MetaQuotes Ltd.

PointsSize (Get method)

곡선을 표시하는 데 사용되는 점의 선형 크기(픽셀)를 반환.

```
int  PointsSize()
```

값 반환

곡선을 정의하는 점의 크기(픽셀).

PointsSize (Set method)

곡선을 표시하는 데 사용되는 점의 선형 크기(픽셀)를 설정.

```
void  PointsSize(
   const int  size      // 점 크기(픽셀)
  \)
```

매개변수

Size

[in] 곡선을 그리는 데 사용되는 점의 선형 크기(픽셀).

© 2000-2025, MetaQuotes Ltd.

PointsFill (Get method)

곡선을 정의하는 점 채우기를 수행할지 여부를 결정하는 플래그를 반환.

```
bool  PointsFill ()
```

값 반환

플래그 값.

참고

true — 채우기 수행

false — 채우기 수행하지 않음

PointsFill (Set method)

곡선을 정의하는 점에 채우기를 수행할지 여부를 결정하는 플래그를 설정.

```
void  PointsFill(
   const bool  fill      // 플래그 값
  \)
```

매개변수

fill

[in] 플래그 값.

참고

true — 채우기 수행

false — 채우기 수행하지 않음

© 2000-2025, MetaQuotes Ltd.

PointsColor (Get method)

점 채우기 색상을 반환.

```
uint  PointsColor ()
```

값 반환

곡선을 정의하는 채우기 점의 색상.

PointsColor (Set method)

점 채우기 색상 설정

```
void  PointsColor(
   const uint  clr      //점 채우기 색상
  \)
```

매개변수

clr

[in] 곡선을 정의하는 채우기 점의 색.

© 2000-2025, MetaQuotes Ltd.

GetX

모든 곡선 점의 X 값을 배열로 가져오기.

```
void  GetX(
   double&  x[]      // X 값 표기 배열
  \)
```

매개변수

x[]

[out] 모든 곡선 점의 X 값을 가져오기 위한 배열.

참고

각 곡선 점은 X 및 Y 값 두 개로 정의됩니다. 이러한 값은 CGraphic 클래스에서 그리기 위한 픽셀 단위의 좌표가 아닙니다.

© 2000-2025, MetaQuotes Ltd.

GetY

모든 곡선 점의 Y 값을 배열로 가져오기.

```
void  GetY(
   double&  y[]      // Y 값을 쓰기 위한 배열
  \)
```

매개변수

y[]

[out] 모든 곡선 점의 Y 값을 가져오기 위한 배열.

참고

각 곡선 점은 X 및 Y 값 두 개로 정의됩니다. 이러한 값은 CGraphic 클래스에서 그리기 위한 픽셀 단위의 좌표가 아닙니다.

© 2000-2025, MetaQuotes Ltd.

LinesStyle (Get method)

선을 사용하여 곡선을 표시할 때 선 스타일을 반환.

```
ENUM_LINE_STYLE  LinesStyle()
```

값 반환

Line style.

LinesStyle (Set method)

선을 사용하여 곡선을 표시할 때 선 스타일을 설정.

```
void  LinesStyle (
   ENUM_LINE_STYLE  style      // 선 스타일
  \)
```

매개변수

스타일

[in] 선 스타일.

© 2000-2025, MetaQuotes Ltd.

LinesIsSmooth (Get method)

곡선을 선별로 표시할 때 평활화를 수행해야 하는지 여부를 정의하는 플래그를 반환.

```
bool  LinesIsSmooth()
```

값 반환

Flag value

참고

true — 평활화 수행

false — 평활화 수행하지 않음

LinesIsSmooth (Set method)

곡선을 선별로 표시할 때 평활화를 수행해야 하는지 여부를 정의하는 플래그를 설정.

```
void  LinesIsSmooth(
   const bool  smooth      // 플래그 값
  \)
```

매개변수

smooth

[in] 플래그 값

참고

true — 평활화 수행

false — 평활화 수행하지 않음

© 2000-2025, MetaQuotes Ltd.

LinesSmoothTension (Get method)

선을 사용하여 그릴 때 곡선 평활화 매개변수를 반환.

```
double  LinesSmoothTension()
```

값 반환

평활화 매개변수 값

참고

'tension' 값이 (0.0; 1.0] 범위 내에 있습니다.

LinesSmoothTension (Set method)

선을 사용하여 그릴 때 곡선 평활화 매개변수를 설정.

```
void  LinesSmoothTension(
   const double  tension      // 매개 변수
  \)
```

매개변수

tension

[in] 평활 매개변수 값.

참고

'tension' 값이 (0.0; 1.0] 범위 내에 있습니다.

© 2000-2025, MetaQuotes Ltd.

LinesSmoothStep (Get method)

선별로 표시할 때 평활화할 근사 선의 길이를 반환.

```
double  LinesSmoothStep()
```

값 반환

근사선의 길이(픽셀).

LinesSmoothStep (Set method)

선별로 표시할 때 평활화할 근사 선의 길이를 설정.

```
void  LinesSmoothStep(
   const double  step      // 선 길이
  \)
```

매개변수

step

[in] 근사선의 길이

© 2000-2025, MetaQuotes Ltd.

LinesEndStyle (Set method)

선을 사용하여 곡선을 표시할 때 선 끝 플로팅 스타일을 나타내는 플래그를 가져오기.

```
ENUM_LINE_END  LinesEndStyle()
```

값 반환

선을 사용하여 곡선을 표시할 때 선 표시 스타일을 나타내는 플래그의 값.

LinesEndStyle (Get method)

선을 사용하여 곡선을 표시할 때 선 끝 플로팅 스타일을 나타내는 플래그를 설정.

```
void  LinesEndStyle(
   ENUM_LINE_END  end_style      // 플래그 값
  \)
```

매개변수

end_style

[in] 선을 사용하여 곡선을 표시할 때 선 끝 플로팅 스타일을 나타내는 플래그 값.

© 2000-2025, MetaQuotes Ltd.

LinesWidth (Get method)

선을 사용하여 곡선을 표시할 때 선 너비 가져오기.

```
int  LinesWidth()
```

값 반환

선 너비.

LinesWidth (Set method)

선을 사용하여 곡선을 표시할 때 선 너비를 설정.

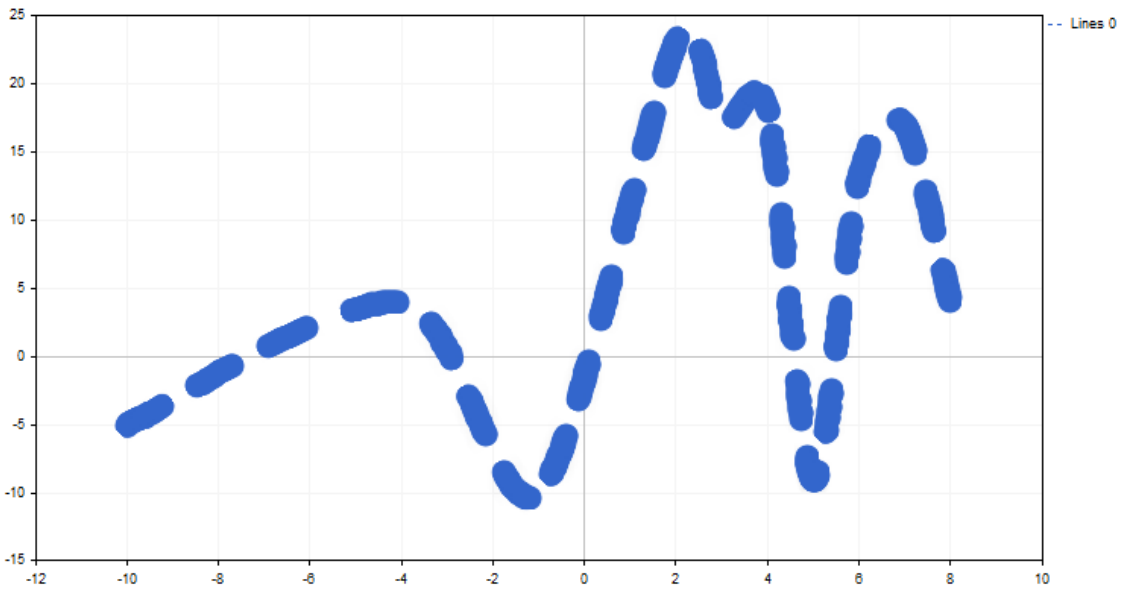
```
void  LinesWidth(
   const int  width      // 선 너비
  \)
```

매개변수

width

[in] 선을 사용하여 곡선을 표시할 때의 선 너비.

예시:



다음 코드를 사용하여 선 너비가 변경되었습니다:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]= { -100,-40,-10,20,30,40,50,60,70,80,120 };
   double y[]= { -5,4,-10,23,17,18,-9,13,17,4,9 };
//--- 그래픽 생성
   CGraphic graphic;
   if(!graphic.Create(0,"ThickLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"ThickLineGraphic");
     }
//--- 곡선 생성
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_LINES);
//--- 곡선 속성 설정
   curve.LinesSmooth(true);
   curve.LinesStyle(STYLE_DASH);
   curve.LinesEndStyle(LINE_END_ROUND);
   curve.LinesWidth(10);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

HistogramWidth (Get method)

히스토그램을 사용하여 표시할 때 열의 너비를 반환.

```
int  HistogramWidth()
```

값 반환

열의 너비(픽셀).

HistogramWidth (Set method)

히스토그램을 사용하여 표시할 때 열의 너비를 설정.

```
void  HistogramWidth(
   const int  width      // 열 너비
  \)
```

매개변수

width

[in] 열 너비((픽셀).

© 2000-2025, MetaQuotes Ltd.

CustomPlotCBData (Get method)

사용자 지정 곡선 플로팅 모드에서 사용할 개체에 대한 포인터를 가져오기.

```
void*  CustomPlotCBData()
```

값 반환

사용자 지정 곡선 플로팅 모드의 개체에 대한 포인터.

CustomPlotCBData (Set method)

사용자 지정 곡선 플로팅 모드에서 사용할 개체에 대한 포인터를 설정하기.

```
void  CustomPlotCBData(
   void*  cbdata      // 개체에 대한 포인터
  \)
```

매개변수

cbdata

[in] 사용자 지정 곡선 플로팅 모드에서 사용할 개체에 대한 포인터

© 2000-2025, MetaQuotes Ltd.

CustomPlotFunction (Get method)

사용자 지정 곡선 플로팅 모드를 구현하는 함수에 대한 포인터를 가져오기.

```
PlotFucntion  CustomPlotFunction()
```

값 반환

사용자 지정 곡선 플로팅 모드를 구현하는 함수에 대한 포인터.

CustomPlotFunction (Set method)

사용자 지정 곡선 플로팅 모드를 구현하는 함수에 대한 포인터를 설정.

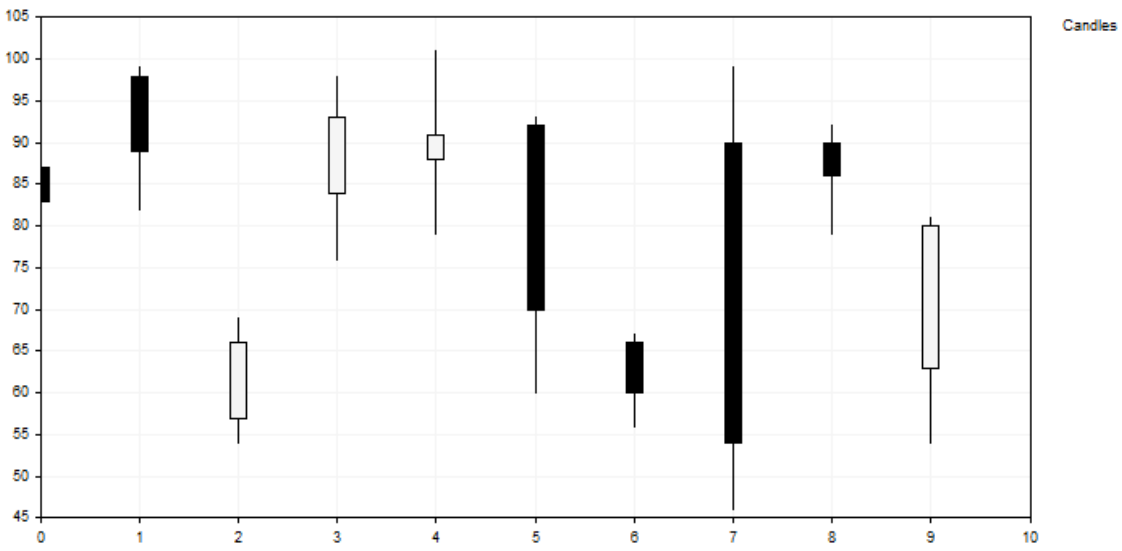
```
void  CustomPlotFunction(
   PlotFucntion  func      // 함수에 대한 포인터
  \)
```

매개변수

func

[in] 사용자 지정 곡선 플로팅 모드를 구현하는 함수에 대한 포인터

예시:



막대로 구성된 이 곡선은 다음 코드를 사용하여 작성됩니다:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| 클래스 CCandle                                                   |
//| 사용: 캔들을 나타내는 클래스                                     |
//+------------------------------------------------------------------+
class CCandle: public CObject
  {
private:
   double            m_open;
   double            m_close;
   double            m_high;
   double            m_low;
   uint              m_clr_inc;
   uint              m_clr_dec;
   int               m_width;

public:
                     CCandle(const double open,const double close,const double high,co
                                                    const int width,const uint clr_
                    ~CCandle(void);
   double            OpenValue(void)            const { return(m_open);      }
   double            CloseValue(void)           const { return(m_close);     }
   double            HigthValue(void)           const { return(m_high);      }
   double            LowValue(void)             const { return(m_low);       }
   uint              CandleColorIncrement(void) const { return(m_clr_inc);   }
   uint              CandleColorDecrement(void) const { return(m_clr_dec);   }
   int               CandleWidth(void)          const { return(m_width);     }
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
CCandle::CCandle(const double open,const double close,const double high,const double l
                                 const int width,const uint clr_inc=0x000000,const uin
                                 m_open(open),m_close(close),m_high(high),m_low(low),
                                 m_clr_inc(clr_inc),m_clr_dec(clr_dec),m_width(width)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CCandle::~CCandle(void)
  {
  }
//+------------------------------------------------------------------+
//| Custom method for plot candles                                   |
//+------------------------------------------------------------------+
void PlotCandles(double &x[],double &y[],int size,CGraphic *graphic,CCanvas *canvas,vo
  {
//--- check obj
   CArrayObj *candles=dynamic_cast<CArrayObj*>(cbdata);
   if(candles==NULL || candles.Total()!=size)
      return;
//--- plot candles
   for(int i=0; i<size; i++)
     {
      CCandle *candle=dynamic_cast<CCandle*>(candles.At(i));
```

© 2000-2025, MetaQuotes Ltd.

```
      if(candle==NULL)
         return;
      //--- 일차 계산
      int xc=graphic.ScaleX(x[i]);
      int width_2=candle.CandleWidth()/2;
      int open=graphic.ScaleY(candle.OpenValue());
      int close=graphic.ScaleY(candle.CloseValue());
      int high=graphic.ScaleY(candle.HigthValue());
      int low=graphic.ScaleY(candle.LowValue());
      uint clr=(open<=close) ? candle.CandleColorIncrement() :  candle.CandleColorDeci
      //--- 캔들 플롯
      canvas.LineVertical(xc,high,low,0x000000);
      //--- 캔들 본체 플롯
      canvas.FillRectangle(xc+width_2,open,xc-width_2,close,clr);
      canvas.Rectangle(xc+width_2,open,xc-width_2,close,0x000000);
     }
  }
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   int count=10;
   int width=10;
   double x[];
   double y[];
   ArrayResize(x,count);
   ArrayResize(y,count);
   CArrayObj candles();
   double max=0;
   double min=0;
//--- 값 생성
   for(int i=0; i<count; i++)
     {
      x[i] = i;
      y[i] = i;
      //--- 값 계산
      double open=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double close=MathRound(50.0+(MathRand()/32767.0)*50.0);
      double high=MathRound(MathMax(open,close)+(MathRand()/32767.0)*10.0);
      double low=MathRound(MathMin(open,close) -(MathRand()/32767.0)*10.0);
      //--- 최대 및 최소 찾기
      if(i==0 || max<high)
         max=high;
      if(i==0 || min>low)
         min=low;
      //--- 캔들 생성
      CCandle *candle=new CCandle(open,close,high,low,width);
      candles.Add(candle);
     }
//--- 그래픽 생성
   CGraphic graphic;
   if(!graphic.Create(0,"CandleGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"CandleGraphic");
     }
//--- 곡선 생성
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_CUSTOM,"Candles");
//--- 곡선 속성 설정
   curve.CustomPlotFunction(PlotCandles);
   curve.CustomPlotCBData(GetPointer(candles));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 그래픽 속성 설정
   graphic.YAxis().Max((int)max);
   graphic.YAxis().Min((int)min);
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

PointsType (Get method)

점선 곡선을 표시할 때 사용되는 점 유형을 가리키는 플래그 가져오기.

```
ENUM_POINT_TYPE  PointsType()
```

값 반환

점 유형을 나타내는 플래그이 값.

PointsType (Set method)

점선 곡선을 표시할 때 사용되는 점 유형을 가리키는 플래그를 설정.

```
void  PointsType(
   ENUM_POINT_TYPE  type      // 플래그 값
  \)
```

매개변수

type

[in] 점선 곡선을 표시할 때 사용되는 점 유형을 가리키는 플래그의 값.

© 2000-2025, MetaQuotes Ltd.

StepsDimension (Get method)

단계 유형 곡선 렌더링에 사용되는 치수를 나타내는 값을 가져오기.

```
int  StepsDimension()
```

값 반환

단계 유형 곡선 렌더링에 사용되는 치수.

StepsDimension (Set method)

단계 유형 곡선 렌더링에 사용되는 치수를 나타내는 값을 설정.

```
void  StepsDimension(
   const int  dimension      // 치수
  \)
```

매개변수

치수

[in] 치수 (0 또는 1).

참고

0 —x (수직선 다음에 따라오는 수평선).

1 —y (수평선 다음에 따라오는 수직선).

© 2000-2025, MetaQuotes Ltd.

TrendLineCoefficients (Get method)

배열에 쓰기 위한 추세선 비율 가져오기.

```
double&  TrendLineCoefficients()
```

값 반환

추세선 비율.

TrendLineCoefficients (Set method)

배열에 쓰기 위한 추세선 비율 설정.

```
void  TrendLineCoefficients(
   double&  coefficients[]      // 표기 비율 배열
  \)
```

매개변수

coefficients[]

[out] 표기 비율 배열.

© 2000-2025, MetaQuotes Ltd.

TrendLineColor (Get method)

곡선에 대한 추세선 색상 가져오기.

```
uint  TrendLineColor()
```

값 반환

추세선 색상.

TrendLineColor (Set method)

곡선의 추세선 색상을 설정.

```
void  TrendLineColor(
   const uint  clr      // 추세선 색상
  \)
```

매개변수

clr

[in] 선 색상

© 2000-2025, MetaQuotes Ltd.

TrendLineVisible (Get method)

추세선 가시성 플래그 가져오기.

```
bool  TrendLineVisible()
```

값 반환

추세선에 표시되는지 여부를 지정하는 플래그 값.

TrendLineVisible (Set 메서드)

추세선 가시성 플래그를 설정.

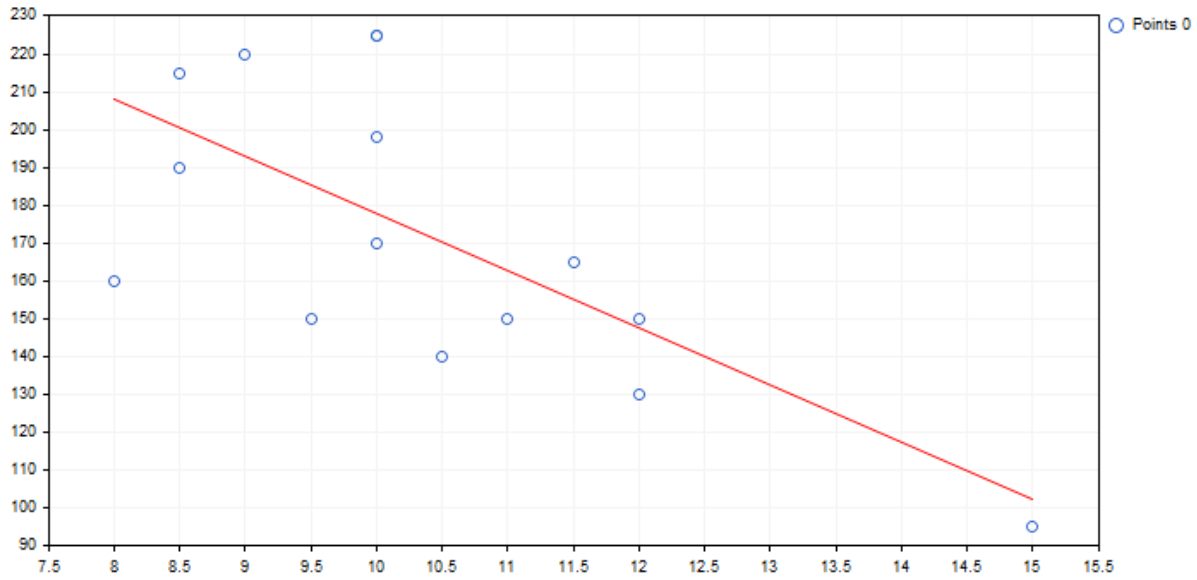
```
void  TrendLineVisible(
   const bool  visible      // 플래그 값
  \)
```

매개변수

visible

[in] 추세선 가시성 플래그 값.

예시:



아래는 차트에 언급된 추세선과 추세선의 코드입니다:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                              TrendLineGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| 스크립트 프로그램 시작 함수                                      |
//+------------------------------------------------------------------+
void OnStart()
  {
   double x[]={12.0,11.5,11.0,12.0,10.5,10.0,9.0,8.5,10.0,8.5,10.0,8.0,9.5,10.0,15.0};
   double y[]={130.0,165.0,150.0,150.0,140.0,198.0,220.0,215.0,225.0,190.0,170.0,160.0
//--- 그래픽 생성
   CGraphic graphic;
   if(!graphic.Create(0,"TrendLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"TrendLineGraphic");
     }
//--- 곡선 생성
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_POINTS);
//--- 곡선 속성 설정
   curve.TrendLineVisible(true);
   curve.TrendLineColor(ColorToARGB(clrRed));
//--- 플롯
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

© 2000-2025, MetaQuotes Ltd.

Update

곡선 좌표를 업데이트.

Y 좌표로 작업하기 위한 버전. 전달된 배열 인덱스는 여기서 X 좌표로 사용됩니다.

```
void  Update(
   const double&  y[]   // Y 좌표
  \)
```

이 버전에서는 X 및 Y 좌표를 사용합니다.

```
void  Update(
   const double&  x[],     // X 좌표
   const double&  y[]      // Y 좌표
  \)
```

CPoint2D 점으로 작업하기 위한 버전.

```
void  Update(
   const CPoint2D&  points[]      // 곡선 좌표
  \)
```

CurveFunction 함수에 대한 포인터로 작업하기 위한 버전.

```
void  Update(
   CurveFunction  function,     // 곡선을 설명하는 함수에 대한 포인터
   const double   from,         // 함수 인수의 초기 값
   const double   to,           // 함수 인수의 최종 값
   const double   step          // 인수 증가
  \)
```

매개변수

x[]
  [in] X 좌표.

y[]
  [in] Y 좌표.

points[]
  [in] 곡선 좌표.

함수
  [in] 곡선을 설명하는 함수에 대한 포인터

from
  [in] 함수 인수의 초기 값

to
  [in] 함수 인수의 끝 값

step
  [in] 인수 증가

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Visible (Get method)

함수가 차트에 표시되는지 여부를 정의하는 플래그 가져오기.

```
void  Visible(
   const bool  visible      //
  \)
```

값 반환

차트에서 함수 가시성을 정의하는 플래그의 값.

Visible (Set method)

함수가 차트에 표시되는지 여부를 정의하는 플래그를 설정.

```
void  Visible(
   const bool  visible      // 플래그 값
  \)
```

매개변수

visible

[in] 차트에서 함수 가시성을 정의하는 플래그 값

© 2000-2025, MetaQuotes Ltd.

CGraphic

CGraphic은 사용자 지정 차트를 만들기 위한 기본 클래스입니다.

설명

CGraphic 클래스는 사용자 지정 차트 작업의 다양한 측면을 제공합니다.

클래스는 주 차트 요소를 저장하고 해당 매개변수를 설정하고 플로팅을 수행합니다.

또한 클래스는 차트의 곡선을 저장하고 다양한 표시 옵션을 제공합니다.

선언

```
class CGraphic
```

제목

```
#include <Graphics\Graphic.mqh>
```

클래스 메서드

메서드	설명
Create	차트 개체에 바인딩된 그래픽 리소스 생성
Delete	차트 제거 및 그래픽 리소스 삭제
Update	구현된 변경 사항 표시
ChartObjectName	차트에 바인딩된 개체의 이름을 가져오기
ResourceName	그래픽 리소스 이름 가져오기
XAxis	X 축에 대한 포인터 가져오기
YAxis	Y 축에 대한 포인터 가져오기
GapSize	차트 요소 사이의 들여쓰기 크기 가져오기/설정하기
BackgroundColor	배경 색상 가져오기/설정하기
BackgroundMain	차트 헤더 가져오기/설정하기
BackgroundMainSize	하위 제목 글꼴 크기 가져오기/설정하기
BackgroundMainColor	차트 헤더 색상 가져오기/설정하기
BackgroundSub	하위 제목 가져오기/설정하기
BackgroundSubSize	하위 제목 글꼴 크기 가져오기/설정하기
BackgroundSubColor	하위 제목 색상 가져오기/설정하기
GridLineColor	그리드 선 색상 가져오기/설정하기
GridBackgroundColor	그리드 배경 색상 가져오기/설정하기

© 2000-2025, MetaQuotes Ltd.

메서드	설명
GridCircleRadius	그리드 노드에서 점 반지름 가져오기/설정하기
GridCircleColor	그리드 노드에서 점 색상 가져오기/설정하기
GridHasCircle	그리드 노드에서 점 플로팅 플래그 가져오기/설정하기
GridAxisLineColor	실수 차트 축 색상 값을 가져오기.
HistoryNameWidth	실수 차트 축 색상 값을 가져오기
HistoryNameSize	곡선 이름의 글꼴 크기 가져오기/설정하기
HistorySymbolSize	표기 규칙 기호 크기 가져오기/설정하기
TextAdd	차트에 텍스트 추가
LineAdd	차트에 선 추가
CurveAdd	차트에 곡선 생성 및 추가
CurvePlot	인덱스로 이전에 생성된 곡선 그림 그리기
CurvePlotAll	이전에 생성된 모든 곡선 그림 표시
CurveGetByIndex	지정된 인덱스로 곡선 가져오기
CurveGetByName	지정된 이름으로 곡선 가져오기
CurveRemoveByIndex	지정된 인덱스로 곡선을 제거.
CurveRemoveByName	지정된 이름으로 곡선을 제거.
CurvesTotal	주어진 차트에 대한 곡선의 수를 가져오기.
MarksToAxisAdd	차트 축에 스케일 표시 추가
MajorMarkSize	차트 축에 스케일 눈금 크기 가져오기/설정학
FontSet	현재 글꼴 매개변수 설정
FontGet	현재 글꼴 매개변수 가져오기
Attach	Get/set a graphical resource and bind it to the그래픽 리소스를 가져오거나 설정하고 CGraphic 클래스 인스턴스에 바인딩
CalculateMaxMinValues	두 축의 최소 및 최대 차트 값을 계산(재계산).
Height	차트 높이(픽셀) 가져오기
IndentDown	아래쪽 테두리에서 차트 들여쓰기 가져오기/설정하기.
IndentLeft	왼쪽 테두리에서 차트 들여쓰기 가져오기/설정하기.
IndentRight	오른쪽 테두리에서 차트 들여쓰기 가져오기/설정하기.
IndentUp	위쪽 테두리에서 들여쓰기 가져오기/설정하기.

© 2000-2025, MetaQuotes Ltd.

メソド	説明
Redraw	차트를 다시 그리기.
ResetParameters	차트 다시 그리기 매개변수를 재설정.
ScaleX	X 축으로 값 스케일 조정.
ScaleY	Y 축으로 값 스케일 조정.
SetDefaultParameters	차트 매개 변수를 기본값으로 설정.
Width	차트 너비(픽셀) 가져오기.

© 2000-2025, MetaQuotes Ltd.

Create

차트 개체에 바인딩된 그래픽 리소스를 생성.

```
bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 하위 창 인덱스
   const int      x1,         // x1 좌표
   const int      y1,         // y1 좌표
   const int      x2,         // x2 좌표
   const int      y2          // y1 좌표
  \)
```

매개변수

차트

[in] 차트 ID.

name

[in] 이름.

subwin

[in] 하위 창 인덱스.

x1

[in] X1 좌표.

y1

[in] Y1 좌표.

x2

[in] X2 좌표.

y2

[in] Y2 좌표.

© 2000-2025, MetaQuotes Ltd.

Delete

차트를 제거하고 그래픽 리소스를 삭제.

```
void  Destroy()
```

© 2000-2025, MetaQuotes Ltd.

Update

구현된 변경 사항 표시.

```
void  Update(
   const bool  redraw=true      // 플래그
  \)
```

매개변수

redraw=true

[in] 플래그 값

© 2000-2025, MetaQuotes Ltd.

ChartObjectName

차트에 바인딩된 개체의 이름 가져오기.

```
string  ChartObjectName()
```

값 반환

차트에 바인딩된 개체의 이름.

© 2000-2025, MetaQuotes Ltd.

ResourceName

그래픽 리소스의 이름 수신

```
string  ResourceName()
```

값 반환

그래픽 리소스의 이름.

© 2000-2025, MetaQuotes Ltd.

XAxis

포인터를 X 축으로 반환.

```
CAxis  *XAxis()
```

값 반환

X 축에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

YAxis

포인터를 Y 축으로 반환.

```
CAxis  *YAxis()
```

값 반환

Y 축에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

GapSize (Get method)

차트 요소 사이의 들여쓰기를 반환.

```
int  GapSize()
```

값 반환

들여쓰기 크기(픽셀).

GapSize (Set method)

차트 요소 사이의 들여쓰기 크기를 설정.

```
void  GapSize(
   const int  size     // 들여쓰기 크기
  \)
```

매개변수

Size

[in] 들여쓰기 크기(픽셀).

© 2000-2025, MetaQuotes Ltd.

BackgroundColor (Get method)

배경 색상 반환.

```
color  BackgroundColor()
```

BackgroundColor (Set method)

배경 색상 설정.

```
void  BackgroundColor(
   const color  clr      // 배경 색상
  \)
```

매개변수

clr

[in] 배경 색상.

© 2000-2025, MetaQuotes Ltd.

BackgroundMain (Get method)

차트 헤더 반환

```
string  BackgroundMain()
```

BackgroundMain (Set method)

차트 헤더 텍스트 설정.

```
void  BackgroundMain(
   const string  main      // 헤더 텍스트
  \)
```

매개변수

main

[in] 차트 헤더 텍스트

© 2000-2025, MetaQuotes Ltd.

BackgroundMainSize (Get method)

헤더 크기 반환.

```
int  BackgroundMainSize()
```

값 반환

헤더 글꼴 크기.

BackgroundMainSize (Set method)

헤더 크기 설정.

```
void  BackgroundMainSize(
   const int  size     // 헤더 크기
  \)
```

매개변수

Size

[in] 헤더 글꼴 크기.

© 2000-2025, MetaQuotes Ltd.

BackgroundMainColor (Get method)

헤더 색상 반환.

```
color  BackgroundMainColor()
```

값 반환

헤더 색상.

BackgroundMainColor (Set method)

헤더 색상 설정.

```
void  BackgroundMainColor(
   const color  clr      // 헤더 색상
  \)
```

매개변수

clr

[in] 헤더 색상.

© 2000-2025, MetaQuotes Ltd.

BackgroundSub (Get method)

하위 제목 반환.

```
string  BackgroundSub()
```

값 반환

하위 제목 텍스트.

BackgroundSub (Set method)

하위 제목 텍스트 설정.

```
void  BackgroundSub (Set method)(
   const string  sub       // 하위 제목 텍스트
  \)
```

매개변수

sub

[in] 하위 제목 텍스트.

© 2000-2025, MetaQuotes Ltd.

BackgroundSubSize (Get method)

하위 제목 글꼴 크기 반환

```
int BackgroundSubSize()
```

BackgroundSubSize (Get method)

하위 제목크기 설정

```
void  BackgroundSubSize(
   const int  size      // 하위 제목 글꼴 크기
  \)
```

매개변수

Size

[in] 하위 제목 글꼴 크기

© 2000-2025, MetaQuotes Ltd.

BackgroundSubColor (Get method)

하위 제목 색상 반환.

```
color  BackgroundSubColor()
```

BackgroundSubColor (Set method)

하위 제목 색상 설정.

```
void  BackgroundSubColor(
   const color  clr      // 하위 제목 색상
  \)
```

매개변수

clr

[in] 하위 제목 색상.

© 2000-2025, MetaQuotes Ltd.

GridLineColor (Get method)

그리드 선 색상을 반환

```
color  GridLineColor()
```

값 반환

그리드 선 색상.

GridLineColor (Set method)

그리드 선 색상을 설정.

```
void  GridLineColor(
   const color  clr      // 선 색상
  \)
```

매개변수

clr

[in] 그리드 선 색상.

© 2000-2025, MetaQuotes Ltd.

GridBackgroundColor (Get method)

그리드 배경 색상 반환

```
color  GridBackgroundColor()
```

값 반환

그리드 배경 색상

GridBackgroundColor (Set method)

그리드 배경 색상 설정

```
void  GridBackgroundColor(
   const color  clr      // 그리드 배경 색상
  \)
```

매개변수

clr

[in] 그리드 배경 색상

© 2000-2025, MetaQuotes Ltd.

GridCircleRadius (Get method)

그리드 노드에서 점의 반지름을 반환.

```
int  GridCircleRadius()
```

값 반환

점 반지름(픽셀).

GridCircleRadius (Set method)

그리드 노드의 점 반지름을 설정

```
void  GridCircleRadius(
   const int  r      // 반지름
  \)
```

매개변수

r

[in] 점 반지름(픽셀).

© 2000-2025, MetaQuotes Ltd.

GridCircleColor (Get method)

그리드 노드의 점 색상 반환.

```
color  GridCircleColor()
```

값 반환

점 색상.

GridCircleColor (Set method)

그리도 노드의 점 색상을 설정.

```
void  GridCircleColor(
   const color  clr      // 점 색상
  \)
```

매개변수

clr

[in] 점 색상.

© 2000-2025, MetaQuotes Ltd.

GridHasCircle (Get method)

그리드 노드의 점을 표시할지 여부를 정의하는 플래그를 반환.

```
bool  GridHasCircle()
```

값 반환

플래그 값.

참고

true — 점을 표시

false — 점을 표시하지 않음

GridHasCircle (Set method)

그리드 노드의 점을 표시할지 여부를 정의하는 플래그를 설정.

```
void  GridHasCircle(
   const bool  has
  \)
```

매개변수

has

[in] 플래그 값.

참고

true — 점을 표시

false — 점을 표시하지 않음

© 2000-2025, MetaQuotes Ltd.

GridAxisLineColor (Get method)

실수 차트 축 색상 값을 가져오기.

```
uint  GridAxisLineColor()
```

값 반환

실수 차트 축 색상.

ㄴGridAxisLineColor (Set 메서드)

실수 차트 축 색상 값을 설정.

```
void  GridAxisLineColor(
   const uint  clr      // 차트 축 색상
  \)
```

매개변수

clr

[in] 실수 차트 축의 색상.

© 2000-2025, MetaQuotes Ltd.

HistoryNameWidth (Get method)

곡선의 이름을 표시하기 위한 최대 허용 길이 반환.

```
int  HistoryNameWidth()
```

값 반환

최대 길이(픽셀).

참고

곡선 이름이 허용된 최대 길이를 초과하면 이름이 잘리고 끝에 점이 투가됩니다.

HistoryNameWidth (Set method)

곡선의 이름을 표시할 수 있는 최대 허용 길이를 설정.

```
void  HistoryNameWidth(
   const int  width      // 최대 길이
  \)
```

매개변수

width

[in] 최대 길이(픽셀).

참고

곡선 이름이 허용된 최대 길이를 초과하면 이름이 잘리고 끝에 점이 투가됩니다.

© 2000-2025, MetaQuotes Ltd.

HistoryNameSize (Get method)

곡선 이름의 글꼴 크기를 반환.

```
int  HistoryNameSize()
```

값 반환

곡선 이름의 글꼴 크기.

HistoryNameSize (Set method)

곡선 이름의 글꼴 크기를 설정.

```
void  HistoryNameSize (Set method)(
   const int  size      // 이름 글꼴 크기
  \)
```

매개변수

Size

[in] 이름 글꼴 크기.

© 2000-2025, MetaQuotes Ltd.

HistorySymbolSize (Get method)

차트의 표기 규칙 기호 크기를 반환

```
int  HistorySymbolSize()
```

값 반환

표기 규칙 기호 크기

HistorySymbolSize (Set method)

차트의 표기 규칙 기호 크기를 설정

```
void  HistorySymbolSize(
   const int  size      // 기호 크기
  \)
```

매개변수

Size

[in] 표기 규칙 기호의 크기.

© 2000-2025, MetaQuotes Ltd.

TextAdd

차트에 텍스트 추가.

X 및 Y 좌표로 작업하기 위한 버전

```
void  TextAdd(
   const int     x,              // X 좌표
   const int     y,              // Y 좌표
   const string  text,           // 텍스트
   const uint    clr,            // 색상
   const uint    alignment=0     // 정렬
  \)
```

CPoint 버전

```
void  TextAdd(
   const CPoint  &point,         // 점 좌표
   const string  text,           // 텍스트
   const uint    clr,            // 색상
   const uint    alignment=0     // 정렬
  \)
```

매개변수

x

[in] X 좌표.

y

[in] Y 좌표.

&point

[in] 점 좌표.

텍스트

[in] 텍스트.

clr

[in] 색상.

alignment=0

[in] 정렬.

© 2000-2025, MetaQuotes Ltd.

LineAdd

차트에 선 추가.

이 버전은 X 및 Y 좌표를 사용합니다

```
void  LineAdd(
   const int   x1,         // x1 좌표
   const int   y1,         // y1 좌표
   const int   x2,         // x2 좌표
   const int   y2,         // y2 좌표
   const uint  clr,        // 색상
   const uint  style       // 스타일
  \)
```

CPoint 버전

```
void  LineAdd2(
   const CPoint  &point1,     // 첫 번째 점 좌표
   const CPoint  &point2,     // 두 번째 점 좌표
   const uint    clr,         // 색상
   const uint    style        // 스타일
  \)
```

매개변수

x1
  [in] X1 좌표.

y1
  [in] Y1 좌표.

x2
  [in] X2 좌표.

y2
  [in] Y2 좌표.

&point1
  [in] 첫 번째 점 좌표.

&point2
  [in] 두 번째 점 좌표.

clr
  [in] 색상.

스타일
  [in] 스타일.

© 2000-2025, MetaQuotes Ltd.

CurveAdd

차트에 새 곡선 작성 및 추가.

이 버전은 Y 좌표를 사용합니다 (곡선 색상이 자동으로 설정됨)

```
CCurve* CurveAdd(
   const double      &y[],          // Y 좌표
   ENUM_CURVE_TYPE   type,          // 곡선 유형
   const string      name=NULL      // 곡선 이름
  \)
```

참고

Y 배열 지수는 곡선의 X 좌표로 사용됩니다.

이 버전은 X 및 Y 좌표를 사용합니다 (곡선 색상이 자동으로 설정됨)

```
CCurve*  CurveAdd(
   const double      &x[],          // X 좌표
   const double      &y[],          // Y 좌표
   ENUM_CURVE_TYPE   type,          // 곡선 유형
   const string      name=NULL      // 곡선 이름
  \)
```

CPoint2D 점으로 작업하기 위한 버전 (곡선 색상이 자동으로 설정됨)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // 점 좌표
   ENUM_CURVE_TYPE   type,          // 곡선 유형
   const string      name=NULL      // 곡선 이름
  \)
```

CurveFunction 함수에 대한 포인터로 작업하기 위한 버전 (곡선 색상이 자동으로 설정됨)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // 함수에 대한 포인터
   const double      from,          // 인수의 초기 값
   const double      to,            // 인수의 최종 값
   const double      step,          // 인수 증가
   ENUM_CURVE_TYPE   type,          // 곡선 유형
   const string      name=NULL      // 곡선 이름
  \)
```

Y 좌표로 작업하기 위한 버전 (곡선 색상은 사용자가 설정)

© 2000-2025, MetaQuotes Ltd.

```
CCurve*  CurveAdd(
   const double      &y[],          // Y 좌표
   const uint        clr,           // 곡선 색상
   ENUM_CURVE_TYPE   type,          // 곡선 유형
   const string      name=NULL      // 곡선 이름
  \)
```

참고

Y 배열 지수는 곡선의 X 좌표로 사용됩니다.

이 버전은 X 및 Y 좌표를 사용합니다 (곡선 색상은 사용자가 설정)

```
CCurve*  CurveAdd(
   const double      &x[],          // X 좌표
   const double      &y[],          // Y 좌표
   const uint        clr,           // 곡선 색상
   ENUM_CURVE_TYPE   type,          // 곡선 유형
   const string      name=NULL      // 곡선 이름
  \)
```

CPoint2D 점으로 작업하기 위한 버전 (곡선 색상은 사용자가 설정)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // 점 좌표
   const uint        clr,           // 곡선 색상
   ENUM_CURVE_TYPE   type,          // 곡선 유형
   const string      name=NULL      // 곡선 이름
  \)
```

CurveFunction 함수에 대한 포인터로 작업하기 위한 버전 (곡선 색상은 사용자 설정)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // 함수에 대한 포인터
   const double      from,          // 인수의 초기 값
   const double      to,            // 인수의 최종 값
   const double      step,          // 인수 증가
   const uint        clr,           // 곡선 색상
   ENUM_CURVE_TYPE   type,          // 곡선 유형
   const string      name=NULL      // 곡선 이름
  \)
```

매개변수

&x[]

© 2000-2025, MetaQuotes Ltd.

[in] X 좌표.

&y[]

[in] Y 좌표.

&points[]

[in] 점의 좌표.

함수

[in] 함수에 대한 포인터.

from

[in] 인수의 초기 값.

to

[in] 인수의 최종 값.

step

[in] 인수 증가.

type

[in] 곡선 유형.

name=NULL

[in] 곡선 이름.

clr

[in] 곡선 색상.

값 반환

작성된 곡선에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

CurvePlot

지정한 인덱스로 이전에 작성된 곡선을 표시.

```
bool  CurvePlot(
   const int  index      // 인덱스
  \)
```

매개변수

인덱스

[in] 곡선 인덱스

값 반환

성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

CurvePlotAll

차트에 이전에 추가된 모든 곡선을 표시.

```
bool  CurvePlotAll()
```

값 반환

성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

CurveGetByIndex

지정된 곡선에 대한 포인터.

```
CCurve*  CurveGetByIndex(
   const int  index      // 곡선 인덱스
  \)
```

매개변수

인덱스

[in] 곡선 인덱스.

값 반환

지정된 인덱스가 있는 곡선에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

CurveGetByName

지정한 이름으로 곡선 가져오기.

```
CCurve*  CurveGetByName(
   const string  name      // 곡선 이름
  \)
```

매개변수

name

[in] 곡선 이름.

값 반환

지정된 이름을 가진 첫 번째 발견된 곡선에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

CurvesTotal

주어진 차트에 대한 곡선의 수를 가져오기.

```
int  CurvesTotal()
```

값 반환

곡선 숫자.

참고

현재 차트의 모든 곡선은 도면 및 가시성 스타일에 관계없이 고려됩니다.

© 2000-2025, MetaQuotes Ltd.

CurveRemoveByName

지정된 이름으로 곡선을 제거.

```
bool  CurveRemoveByName(
   const string  name      // 곡선 이름
  \)
```

매개변수

name

[in] 제거할 곡선 이름.

값 반환

성공하면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CurveRemoveByIndex

지정된 인덱스로 곡선을 제거.

```
bool  CurveRemoveByIndex(
   const int  index      // 곡선 인덱스
  \)
```

매개변수

인덱스

[in] 제거할 곡선의 인덱스.

값 반환

성공하면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

MarksToAxisAdd

지정된 차트 축에 척도 표시(틱)를 추가.

```
bool  MarksToAxisAdd(
   const double         &marks[],        // 틱 좌표
   const int            mark_size,       // 틱 크기
   ENUM_MARK_POSITION   position,        // 틱 위치
   const int            dimension=0      // 치수
  \)
```

매개변수

&marks[]

[in] 틱 좌표

mark_size

[in] 틱 크기

포지션

[in] 틱 위치

dimension=0

[in] 0 — X 축에 추가,

1 — Y 축에 추가

값 반환

성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

MajorMarkSize (Get method)

좌표 축에서 눈금 크기를 반환.

```
int  MajorMarkSize()
```

MajorMarkSize (Set method)

좌표 축에서 눈금 크기를 반환.

```
void  MajorMarkSize(
   const int  size      // 틱 크기
  \)
```

매개변수

Size

[in] 틱 크기(픽셀).

© 2000-2025, MetaQuotes Ltd.

FontSet

현재 글꼴 매개변수를 설정.

```
bool  FontSet(
   const string  name,        // 이름
   const int     size,        // 크기
   const uint    flags=0,     // 플래그
   const uint    angle=0      // 각도
  \)
```

매개변수

name

[in] 이름.

Size

[in] Size.

flags=0

[in] Flags.

angle=0

[in] Angle.

값 반환

성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

FontGet

현재 글꼴 매개변수 가져오기.

```
void  FontGet(
   string  &name,      // 이름
   int     &size,      // 크기
   uint    &flags,     // 플래그
   uint    &angle      // 각도
  \)
```

매개변수

&name

[out] 이름.

&size

[out] 크기.

&flags

[out] 플래그.

&angle

[out] 각도.

© 2000-2025, MetaQuotes Ltd.

Attach

OBJ_BITMAP_LABEL 개체에서 그래필 리소스를 가져와 CGraphic 클래스 인스턴스에 바인딩하기 위한 버전:

```
bool  Attach(
   const long     chart_id,      // 차트 ID
   const string   objname        // 그래픽 개체 이름
  \)
```

OBJ_BITMAP_LABEL 개체에 대한 그래픽 리소스를 만들고 CGraphic 클래스 인스턴스에 바인딩하기 위한 버전:

```
bool  Attach(
   const long     chart_id,      // 차트 ID
   const string   objname,       // 그래픽 개체 이름
   const int      width,         // 이미지 너비
   const int      height         // 이미지 높이
  \)
```

매개변수

chart_id

[in] 차트 ID.

objname

[in] 그래픽 개체의 이름.

width

[in] 리소스 이미지 너비.

height

[in] 리소스 이미지 높이.

값 반환

성공하면 true, 개체 바인딩에 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

CalculateMaxMinValues

두 축의 최소 및 최대 차트 값을 계산(재계산).

```
void  CalculateMaxMinValues()
```

© 2000-2025, MetaQuotes Ltd.

Height

차트 높이(픽셀) 가져오기

```
int  Height()
```

값 반환

차트 높이(픽셀).

© 2000-2025, MetaQuotes Ltd.

IndentDown (Get method)

아래쪽 테두리에서 차트 들여쓰기 가져오기.

```
int  IndentDown()
```

값 반환

들여쓰기 크기(픽셀).

IndentDown (Set method)

아래쪽 테두리에서 차트 들여쓰기 설정하기.

```
void  IndentDown(
   const int  down      // 들여쓰기 크기
  \)
```

매개변수

down

[in] 들여쓰기 크기(픽셀).

© 2000-2025, MetaQuotes Ltd.

IndentLeft (Get method)

왼쪽 테두리에서 차트 들여쓰기 가져오기.

```
int  IndentLeft()
```

값 반환

들여쓰기 크기(픽셀).

IndentLeft (Set method)

왼쪽 테두리에서 차트 들여쓰기 설정하기.

```
void  IndentLeft(
   const int  left      // 들여쓰기 크기
  \)
```

매개변수

left

[in] 들여쓰기 크기(픽셀).

© 2000-2025, MetaQuotes Ltd.

IndentRight (Get method)

오른쪽 테두리에서 차트 들여쓰기 가져오기.

```
int  IndentRight()
```

값 반환

들여쓰기 크기(픽셀).

IndentRight (Set method)

오른쪽 테두리에서 차트 들여쓰기 설정하기.

```
void  IndentRight(
   const int  right      // 들여쓰기 크기
  \)
```

매개변수

right

[in] 들여쓰기 크기(픽셀).

© 2000-2025, MetaQuotes Ltd.

IndentUp (Get method)

위쪽 테두리에서 들여쓰기 가져오기.

```
int  IndentUp()
```

값 반환

들여쓰기 크기(픽셀).

IndentUp (Set method)

위쪽 테두리에서 들여쓰기 설정하기.

```
void  IndentUp(
   const int  up      // 들여쓰기 크기
  \)
```

매개변수

up

[in] 들여쓰기 값(픽셀).

© 2000-2025, MetaQuotes Ltd.

Redraw

차트를 다시 그리기.

```
bool  Redraw(
   const bool  rescale=false      // 플래그 값
  \)
```

매개변수

rescale=false

[in] 차트 스케일을 다시 조정해야 하는지 여부를 나타내는 플래그.

값 반환

성공하면 true, 실패하면 false.

© 2000-2025, MetaQuotes Ltd.

ResetParameters

차트 다시 그리기 매개변수를 재설정.

```
void  ResetParameters()
```

© 2000-2025, MetaQuotes Ltd.

ScaleX

X 축으로 값 스케일 조정.

```
virtual int  ScaleX(
   double  x      // X 축 값
  \)
```

매개변수

x

[in] X 축 실수 값.

값 반환

값(픽셀).

참고

차트에 표시하기 위해 실제 값을 픽셀로 조정.

© 2000-2025, MetaQuotes Ltd.

ScaleY

Y 축으로 값 스케일 조정.

```
virtual int  ScaleY(
   double  y      // Y 축 값
  \)
```

매개변수

y

[in] Y 축 실수 값.

값 반환

값(픽셀).

참고

차트에 표시하기 위해 실제 값을 픽셀로 조정.

© 2000-2025, MetaQuotes Ltd.

SetDefaultParameters

차트 매개 변수를 기본값으로 설정.

```
void  SetDefaultParameters()
```

© 2000-2025, MetaQuotes Ltd.

Width

차트 너비(픽셀) 가져오기.

```
int  Width()
```

값 반환

차트 너비(픽셀).

© 2000-2025, MetaQuotes Ltd.

기술 지표 및 시계열

이 섹션에서는 기술 지표 및 시계열 클래스의 기술 세부 정보와 표준 MQL5 라이브러리의 해당 구성 요소에 대한 설명이 포함되어 있습니다.

기술 지표 및 시계열 클래스를 사용하면 애플리케이션(스크립트, 엑스퍼트 어드바이저) 개발 시간을 절약할 수 있습니다.

MQL5 스탠다드 라이브러리(기술 지표 및 시계열의 경우)는 Include₩Indicators 폴더의 너미널 작업 디렉토리에 있습니다.

Class/group	설명
기본 클래스	기본 및 보조 클래스 그룹
시계열 클래스	시계열 클래스 그룹
추세 지표	추세 지표 클래스 그룹
오실레이터	오실레이터 지표 클래스 그룹
볼륨 지표	볼륨 지표 클래스 그룹
빌 윌리엄즈(Bill Williams) 지표	빌 윌리엄즈 지표 클래스 그룹
사용자 지정 지표	사용자 지정 지표 클래스

© 2000-2025, MetaQuotes Ltd.

기본 및 보조 기술 지표 및 시계열 클래스

이 섹션에는 기본 및 보조 기술 지표 및 시계열 클래스의 기술 세부 정보와 표준 MQL5 라이브러리의 해당 구성요소에 대한 설명이 포함되어 있습니다.

클래스/그룹	Description
CSpreadBuffer	내역 분산 버퍼 클래스
CTimeBuffer	과거 개시 가격 버퍼 클래스
CTickVolumeBuffer	내역 틱 볼륨 버퍼 클래스
CRealVolumeBuffer	내역 실제 볼륨 버퍼 클래스
CDoubleBuffer	double 형식 데이터 버퍼의 기본 클래스
COpenBuffer	시가 막대 버퍼 클래스
CHighBuffer	고가 막대 버퍼 클래스
CLowBuffer	저가 막대 버퍼 클래스
CCloseBuffer	종가 막대 버퍼 클래스
CIndicatorBuffer	기술 지표 버퍼 클래스
CSeries	시계열 데이터에 액세스하기 위한 기본 클래스
CPriceSeries	가격 데이터에 대한 액세스를 위한 기본 클래스
CIndicator	기술 지표의 기본 클래스
CIndicators	기술 지표 및 시계열 모음

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

CSpreadBuffer

CSpreadBuffer는 역사상 막대의 스프레드에 대한 단순화된 액세스를 위한 클래스입니다.

Description

CSpreadBuffer 클래스는 과거의 막대의 스프레드에 대한 단순한 액세스를 제공합니다.

Declaration

```
class CSpreadBuffer: public CArrayInt
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
  CArray
    CArrayInt
      CSpreadBuffer

그룹별 클래스 메서드

Attributes	
Size	버퍼 사이즈 설정
Settings	
SetSymbolPeriod	심볼 및 주기 설정
데이터 액세스	
At	버퍼 요소 가져오기
데이타 업데이트	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayInt 클래스에서 상속된 메서드

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

버퍼 사이즈를 설정.

```
void  Size(
   const int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

심볼 및 주기 설정.

```
void  SetSymbolPeriod(
   const string           symbol,     // 심볼
   const ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 새 심볼.

period

[in] 새 심볼(ENUM_TIMEFRAMES 열거값).

© 2000-2025, MetaQuotes Ltd.

At

버퍼 요소 가져오기.

```
int  At(
   const int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 색인.

Return Value

특정 색인을 가진 버퍼 요소.

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CTimeBuffer

CTimeBuffer는 역사상 술집 개점 시간에 대한 단순화된 액세스를 위한 클래스입니다.

Description

CTimeBuffer 클래스는 과거의 막대의 시작 시간에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class CTimeBuffer: public CArrayLong
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayLong
CTimeBuffer

그룹별 클래스 메서드

Attributes	
Size	버퍼 사이즈 설정
Settings	
SetSymbolPeriod	심볼 및 주기 설정
데이터 액세스 메서드	
At	색인별로 버퍼 요소를 가져옵니다
데이타 업데이트 메서드	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayLong 클래스에서 상속된 메서드

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

버퍼 사이즈를 설정.

```
void  Size(
   const int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

심볼 및 주기 설정.

```
void  SetSymbolPeriod(
   const string           symbol,     // 심볼
   const ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 새 심볼.

period

[in] 새 심볼(ENUM_TIMEFRAMES 열거값).

© 2000-2025, MetaQuotes Ltd.

At

버퍼 요소 가져오기.

```
long  At(
   const int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 색인.

Return Value

특정 색인을 가진 버퍼 요소.

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CTickVolumeBuffer

CTickVolumeBuffer는 과거 막대의 틱 볼륨에 대한 단순 액세스를 위한 클래스입니다.

Description

CTickVolumeBuffer 클래스는 과거 막대의 틱 볼륨에 대한 단순 액세스를 제공합니다.

Declaration

```
class CTickVolumeBuffer: public CArrayLong
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayLong
CTickVolumeBuffer

그룹별 클래스 메서드

Attributes	
Size	버퍼 사이즈 설정
Settings	
SetSymbolPeriod	심볼 및 주기 설정
데이터 액세스 메서드	
At	색인별로 버퍼 요소를 가져옵니다
데이타 업데이트 메서드	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayLong 클래스에서 상속된 메서드

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

버퍼 사이즈를 설정.

```
void  Size(
   const int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

심볼 및 주기 설정.

```
void  SetSymbolPeriod(
   const string           symbol,     // 심볼
   const ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 새 심볼.

period

[in] 새 심볼(ENUM_TIMEFRAMES 열거값).

© 2000-2025, MetaQuotes Ltd.

At

버퍼 요소 가져오기.

```
long  At(
   const int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 색인.

Return Value

특정 색인을 가진 버퍼 요소.

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CRealVolumeBuffer

CRealVolumeBuffer는 과거의 실제 볼륨 막대에 대한 단순화된 액세스를 위한 클래스입니다.

Description

CTickVolumeBuffer 클래스는 과거의 실제 볼륨 막대에 대한 단순 액세스를 제공합니다.

Declaration

```
class CRealVolumeBuffer: public CArrayLong
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayLong
CRealVolumeBuffer

그룹별 클래스 메서드

Attributes	
Size	버퍼 사이즈 설정
Settings	
SetSymbolPeriod	심볼 및 주기 설정
데이터 액세스	
At	버퍼 요소 가져오기
데이타 업데이트	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayLong 클래스에서 상속된 메서드

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

버퍼 사이즈를 설정.

```
void  Size(
   const int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

심볼 및 주기 설정.

```
void  SetSymbolPeriod(
   const string           symbol,     // 심볼
   const ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 새 심볼.

period

[in] 새 심볼(ENUM_TIMEFRAMES 열거값).

© 2000-2025, MetaQuotes Ltd.

At

버퍼 요소 가져오기.

```
long  At(
   const int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 색인.

Return Value

특정 색인을 가진 버퍼 요소.

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CDoubleBuffer

CDoubleBuffer는 더블 타입의 데이터 버퍼에 대한 단순화된 접근을 위한 기본 클래스입니다.

Description

CDoubleBuffer 클래스는 double 타입의 버퍼 데이터에 대한 단순한 접근을 제공합니다.

Declaration

```
class CDoubleBuffer: public CArrayDouble
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
  CArray
    CArrayDouble
      CDoubleBuffer

직계 종속

CCloseBuffer, CHighBuffer, CIndicatorBuffer, CLowBuffer, COpenBuffer

그룹별 클래스 메서드

Attributes	
Size	버퍼 사이즈 설정
Settings	
SetSymbolPeriod	심볼 및 주기 설정
데이터 액세스	
At	버퍼 요소 가져오기
데이타 업데이트	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayDouble 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

© 2000-2025, MetaQuotes Ltd.

Size

버퍼 사이즈를 설정.

```
void  Size(
   const int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

SetSymbolPeriod

심볼 및 주기 설정.

```
void  SetSymbolPeriod(
   const string           symbol,     // 심볼
   const ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 새 심볼.

period

[in] 새 심볼(ENUM_TIMEFRAMES 열거값).

© 2000-2025, MetaQuotes Ltd.

At

버퍼 요소 가져오기.

```
double  At(
   const int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 색인.

Return Value

특정 색인을 가진 버퍼 요소.

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

COpenBuffer

COpenBuffer는 역사상 술집 오픈 가격에 대한 단순화된 액세스를 위한 클래스입니다.

Description

COpenBuffer 클래스는 과거 바의 공개 가격에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class COpenBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayDouble
CDoubleBuffer
COpenBuffer

그룹별 클래스 메서드

데이타 업데이트	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayDouble 클래스에서 상속된 메서드

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

CDoubleBuffer 클래스에서 상속된 메서드

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CHighBuffer

CHighBuffer는 역사상 가장 비싼 술집을 간단하게 이용할 수 있는 클래스입니다.

Description

CHighBuffer 클래스는 역사상 가장 높은 바 가격에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class CHighBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayDouble
CDoubleBuffer
CHighBuffer

그룹별 클래스 메서드

데이타 업데이트	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayDouble 클래스에서 상속된 메서드

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

CDoubleBuffer 클래스에서 상속된 메서드

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CLowBuffer

CLowBuffer는 역사상 가장 낮은 바 가격에 대한 단순화된 액세스를 위한 클래스입니다.

Description

CLowBuffer 클래스는 역사상 가장 저렴한 바 가격에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class CLowBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayDouble
CDoubleBuffer
CLowBuffer

그룹별 클래스 메서드

데이타 업데이트	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayDouble 클래스에서 상속된 메서드

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

CDoubleBuffer 클래스에서 상속된 메서드

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CCloseBuffer

CCloseBuffer는 역사상 가장 가까운 바 가격에 대한 단순화된 접근을 위한 클래스입니다.

Description

CCloseBuffer 클래스는 역사상 가장 가까운 바 가격에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class CCloseBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CCloseBuffer

그룹별 클래스 메서드

데이타 업데이트	
virtual Refresh	버퍼 업데이트
virtual RefreshCurrent	현재 값 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayDouble 클래스에서 상속된 메서드

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

CDoubleBuffer 클래스에서 상속된 메서드

Size, At, SetSymbolPeriod

© 2000-2025, MetaQuotes Ltd.

Refresh

버퍼 업데이트

```
virtual bool  Refresh()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

버퍼의 현재(0번째) 요소를 업데이트합니다.

```
virtual bool  RefreshCurrent()
```

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CIndicatorBuffer

CIndicatorBuffer는 지표의 버퍼 데이터에 대한 단순 액세스를 위한 클래스입니다.

Description

CIndicatorBuffer 클래스는 기술 지표의 데이터 버퍼에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class CIndicatorBuffer: public CDoubleBuffer
```

Title

```
#include <Indicators\Indicator.mqh>
```

상속 계층

CObject
CArray
CArrayDouble
CDoubleBuffer
CIndicatorBuffer

그룹별 클래스 메서드

Attributes	
Offset	버퍼의 간격띄우기를 가져오거나 설정합니다.
Name	버퍼명을 가져오거나 설정합니다.
데이터 액세스	
At	버퍼의 요소를 가져옵니다
데이타 업데이트	
Refresh	버퍼 업데이트
RefreshCurrent	현재 값만 업데이트합니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayDouble 클래스에서 상속된 메서드

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

CDoubleBuffer 클래스에서 상속된 메서드

Size, At, SetSymbolPeriod

© 2000–2025, MetaQuotes Ltd.

Offset

버퍼의 오프셋을 가져옵니다.

```
int  Offset() const
```

Return Value

버퍼 오프셋.

Offset

버퍼의 오프셋을 설정합니다.

```
void  Offset(
   const int  offset      // 오프셋
   )
```

Parameters

offset

[in] 새 버퍼 오프셋.

© 2000-2025, MetaQuotes Ltd.

Name

버퍼의 이름을 가져옵니다.

```
string  Name() const
```

Return Value

버퍼의 이름.

Name

버퍼의 이름을 설정합니다.

```
void  Name(
   const string  name      // 이름
   )
```

Parameters

name

[in] 버퍼의 새 이름.

© 2000-2025, MetaQuotes Ltd.

At

버퍼 요소를 가져옵니다.

```
double  At(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 색인.

Return Value

특정 색인을 가진 버퍼 요소.

© 2000-2025, MetaQuotes Ltd.

Refresh

전체 버퍼를 업데이트합니다.

```
bool  Refresh(
   const int  handle,     // 지표 핸들
   const int  num         // 버퍼 수
   )
```

Parameters

handle

[in] 지표 핸들.

num

[in] 지표의 버퍼 인덱스.

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

현재(0번째) 버퍼 요소를 업데이트합니다.

```
bool  RefreshCurrent(
   const int  handle,     // 지표의 핸들
   const int  num         // 버퍼 수
   )
```

Parameters

handle

[in] 지표 핸들.

num

[in] 지표 버퍼 수.

Return Value

true – 성공, false – 버퍼 업데이트 불가.

© 2000-2025, MetaQuotes Ltd.

CSeries

CSeries는 표준 라이브러리의 시계열 데이터에 대한 액세스를 위한 기본 클래스입니다.

Description

CSeries 클래스는 모든 하위 항목(시계 및 지표 클래스)에 대한 시리즈 데이터 작업과 관련된 모든 MQL5 API 일반 기능에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class CSeries: public CArrayObj
```

Title

```
#include <Indicators\Series.mqh>
```

상속 계층

CObject
  CArray
    CArrayObj
      CSeries

직계 종속

CIndicator, CiRealVolume, CiSpread, CiTickVolume, CiTime, CPriceSeries

그룹별 클래스 메서드

Attributes	
Name	시계열 또는 지표의 이름을 가져옵니다
BuffersTotal	시계열 또는 지표의 버퍼 수를 가져옵니다
Timeframe	시계열 또는 지표의 시간 플래그를 가져옵니다
Symbol	시계열 또는 지표의 심볼을 가져옵니다
Period	시계열 또는 지표의 기간을 가져옵니다
RefreshCurrent	현재 데이터 업데이트 플래그를 설정/재설정합니다
데이터 액세스	
virtual BufferResize	시계열 또는 지표의 버퍼 크기를 설정합니다
데이타 업데이트	
virtual Refresh	시계열 또는 지표 데이터 업데이트
PeriodDescription	ENUM_TIMEFRAMES을 문자열로 변환합니다

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

Name

시계열 또는 지표의 이름을 가져옵니다.

```
string  Name() const
```

Return Value

시계열 또는 지표의 이름.

© 2000-2025, MetaQuotes Ltd.

BuffersTotal

시계열 또는 지표의 버퍼 수를 가져옵니다.

```
int  BuffersTotal() const
```

Return Value

시계열 또는 지표의 버퍼 수

Note

시계열이 단 하나의 버퍼만 가지고 있습니다.

© 2000-2025, MetaQuotes Ltd.

Timeframe

시계열 또는 지표의 시간 플래그를 가져옵니다.

```
int  Timeframe() const
```

Return Value

시계열 또는 지표의 타임프레임 플래그.

Note

개체 가시성 플래그로 생성됩니다.

© 2000-2025, MetaQuotes Ltd.

Symbol

시계열 또는 지표의 기호를 가져옵니다.

```
string  Symbol() const
```

Return Value

시계열 또는 지표의 기호

© 2000-2025, MetaQuotes Ltd.

Period

시계열 또는 지표의 주기를 가져옵니다.

```
ENUM_TIMEFRAMES  Period() const
```

Return Value

시계열 또는 지표의 주기(ENUM_TIMEFRAMES 열거 값).

© 2000-2025, MetaQuotes Ltd.

RefreshCurrent

시계열 또는 지표의 현재 값을 지속적으로 업데이트하도록 플래그를 설정합니다.

```
string  RefreshCurrent(
   const bool  flag      // 값
   )
```

Parameters

flag

[in] 새 플래그.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

BufferSize

시계열 버퍼 또는 지표 버퍼에서 사용할 수 있는 데이터 양을 반환합니다.

```
int  BufferSize() const
```

Return Value

시계열 버퍼 또는 지표 버퍼에서 사용할 수 있는 데이터 양.

© 2000-2025, MetaQuotes Ltd.

BufferResize

시계열 또는 지표의 버퍼 크기를 설정합니다.

```
virtual bool  BufferResize(
   const int  size      // 사이즈
   )
```

Parameters

size

[in] 버퍼의 새 크기.

Return Value

true – 성공, 그렇지 않으면 – false.

Note

모든 시계열 또는 지표 버퍼의 크기가 동일합니다.

© 2000-2025, MetaQuotes Ltd.

Refresh

시계열 또는 지표의 데이터를 업데이트합니다.

```
virtual void  Refresh(
   const int  flags      // 플래그
   )
```

Parameters

flags

[in] 업데이트할 타임프레임(플래그).

© 2000-2025, MetaQuotes Ltd.

PeriodDescription

지정한 ENUM_TIMEFRAMES 열거형의 문자열 표현을 가져옵니다.

```
string  PeriodDescription(
   const int  val=0      // 값
   )
```

Parameters

val=0

[in] 전환할 값.

Return Value

특정 ENUM_TIMEFRAMES 열거의 문자열 표현.

Note

값이 지정되지 않았거나 0과 같으면 시계열 또는 지표의 시간이 문자열로 변환됩니다.

© 2000-2025, MetaQuotes Ltd.

CPriceSeries

CPriceSeries는 가격 데이터에 액세스하기 위한 기본 클래스입니다.

Description

Cseries 클래스는 모든 하위 항목에 대한 가격 데이터 작업을 위한 MQL5 API 일반 기능에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class CPriceSeries: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries

직계 종속

CiClose, CiHigh, CiLow, CiOpen

그룹별 클래스 메서드

Create	
virtual BufferResize	시리즈 버퍼의 사이즈를 설정합니다
데이터 액세스	
virtual GetData	지정된 시계열 버퍼 요소를 가져옵니다
데이타 업데이트	
virtual Refresh	시계열 데이터 업데이트
극단적 값 탐색	
virtual MinIndex	지정된 범위에서 최소 값의 색인을 가져옵니다
virtual MinValue	지정된 범위의 최소값을 가져옵니다
virtual MaxIndex	지정된 범위에서 최대값 색인을 가져옵니다
virtual MaxValue	지정된 범위의 최대값을 가져옵니다

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

BufferResize

시리즈 버퍼 크기를 설정합니다.

```
virtual void  BufferResize(
   const int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

Return Value

true - 성공, 그렇지 않으면 - false.

© 2000-2025, MetaQuotes Ltd.

GetData

지정된 시계열 버퍼 요소를 가져옵니다.

```
double  GetData(
   const int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 색인.

Return Value

시리즈 버퍼 요소, 또는 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Refresh

시계열 데이터 업데이트

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // 플래그
   )
```

Parameters

flags=OBJ_ALL_PERIODS

[in] 타임프레임 플래그.

© 2000-2025, MetaQuotes Ltd.

MinIndex

지정된 범위에서 최소 값의 색인을 가져옵니다.

```
virtual int  MinIndex(
   const int  start,     // 크기
   const int  count      // 수
   ) const
```

Parameters

start

[in] 검색 범위 초기 인덱스.

count

[in] 검색 범위 크기(요소 수).

Return Value

지정된 범위에서 직렬 버퍼의 최소값 인덱스 또는 -1.

© 2000-2025, MetaQuotes Ltd.

MinValue

지정된 범위의 최소값을 가져옵니다.

```
virtual double  MinValue(
   const int   start,     // 크기
   const int   count,     // 수
   int&        index      // 참조
   ) const
```

Parameters

start

[in] 검색 범위 초기 인덱스.

count

[in] 검색 범위 크기(요소 수).

index

[out] 찾은 요소의 색인 값을 배치하기 위한 변수에 대한 참조.

Return Value

지정된 범위에 있는 영상 시리즈 버퍼의 최소값 또는 EMPTY_VALUE.

Note

찾은 요소의 인덱스는 인덱스 참조에 의해 저장됩니다.

© 2000-2025, MetaQuotes Ltd.

MaxIndex

지정된 범위에서 최대값 색인을 가져옵니다.

```
virtual int  MaxIndex(
   const int  start,     // 색인
   const int  count      // 수
   ) const
```

Parameters

start

[in] 검색 범위 초기 인덱스.

count

[in] 검색 범위 크기(요소 수).

Return Value

지정된 범위에서 직렬 버퍼의 최대값 인덱스 또는 -1.

© 2000-2025, MetaQuotes Ltd.

MaxValue

지정된 범위의 최대값을 가져옵니다.

```
virtual double  MaxValue(
   const int    start,     // 크기
   const int    count,     // 양
   int&         index      // 참조
   ) const
```

Parameters

start

[in] 검색 범위 초기 인덱스.

count

[in] 검색 범위 크기(요소 수).

index

[out] 찾은 요소의 색인 값을 배치하기 위한 변수에 대한 참조.

Return Value

지정된 범위에서 직렬 버퍼의 최대값 또는 EMPTY_VALUE.

Note

찾은 요소의 인덱스는 인덱스 참조에 의해 저장됩니다.

© 2000-2025, MetaQuotes Ltd.

CIndicator

CIndicator는 MQL5 표준 라이브러리의 기술 지표 클래스를 위한 기본 클래스입니다.

Description

CIndicator 클래스는 모든 하위 항목에 대해 일반 MQL5 API 기술 지표 기능에 대한 단순화된 액세스를 제공합니다.

Declaration

```
class CIndicator: public CSeries
```

Title

```
#include <Indicators\Indicator.mqh>
```

상속 계층

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator

직계 종속

CiAC, CiAD, CiADX, CiADXWilder, CiAlligator, CiAMA, CiAO, CiATR, CiBands, CiBearsPower, CiBullsPower, CiBWMFI, CiCCI, CiChaikin, CiCustom, CiDEMA, CiDeMarker, CiEnvelopes, CiForce, CiFractals, CiFrAMA, CiGator, CiIchimoku, CiMA, CiMACD, CiMFI, CiMomentum, CiOBV, CiOsMA, CiRSI, CiRVI, CiSAR, CiStdDev, CiStochastic, CiTEMA, CiTriX, CiVIDyA, CiVolumes, CiWPR

그룹별 클래스 메서드

Attributes	
Handle	지표의 핸들을 가져옵니다.
Status	지표의 상태를 가져옵니다.
FullRelease	핸들을 해제하는 플래그를 설정합니다.
Creation	
Create	지표를 작성합니다
BufferResize	새 버퍼 크기를 설정합니다
데이터 액세스	
GetData	지표 버퍼에서 데이터를 복사합니다
데이타 업데이트 메서드	
Refresh	지표 데이터를 업데이트합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
최소/최대값 찾기	
Minimum	지정된 범위에서 최소 값의 색인을 가져옵니다.
MinValue	지정된 범위의 최소값을 가져옵니다.
Maximum	지정된 범위에서 최대값 색인을 가져옵니다.
MaxValue	지정된 범위에서 최대값을 가져옵니다.
열거 전환	
MethodDescription	ENUM_MA_METHOD를 문자열로 변환합니다
PriceDescription	ENUM_APPLIED_PRICE를 문자열로 변환합니다
VolumeDescription	ENUM_APPLIED_VOLUME를 문자열로 변환합니다
차트 작업	
AddToChart	차트에 지표를 추가합니다.
DeleteFromChart	차트에서 지표를 삭제합니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Handle

지표의 핸들을 가져옵니다.

```
int  Handle() const
```

Return Value

지표 핸들.

© 2000-2025, MetaQuotes Ltd.

Status

지표의 상태를 가져옵니다.

```
string  Status() const
```

Return Value

지표 생성 상태.

© 2000-2025, MetaQuotes Ltd.

FullRelease

핸들을 해제하는 플래그를 설정합니다.

```
void  FullRelease(
   const bool  flag=true      // 플래그
   )
```

Parameters

flag

[in] 핸들 해제 플래그의 새 값.

© 2000-2025, MetaQuotes Ltd.

Create

지정한 매개 변수를 사용하여 지정된 유형의 지표를 작성합니다.

```
bool  Create(
   const string           symbol,          // 심볼
   const ENUM_TIMEFRAMES  period,          // 주기
   const ENUM_INDICATOR   type,            // 유형
   const int              num_params,      // 매개변수 수
   const MqlParam&        params[]         // 매개변수 배열
   )
```

Parameters

symbol

[in] 지표 심볼.

period

[in] 지표 타임프레임(ENUM_TIMEFRAMES 열거값).

type

[in] 지표 유형(ENUM_INDICATOR 열거값).

num_params

[in] 지표의 모수 수.

params

[in] 지표의 매개변수 배열 참조.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

BufferResize

지표 버퍼의 크기를 설정합니다.

```
virtual bool  BufferResize(
   const int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

Return Value

true – 성공, 그렇지 않으면 – false.

Note

모든 지표 버퍼의 크기가 동일합니다.

© 2000-2025, MetaQuotes Ltd.

BarsCalculated

지표에 대해 계산된 데이터 양을 반환합니다.

```
int  BarsCalculated() const;
```

Return Value

지표 버퍼에서 계산된 데이터 양을 반환하고, 오류가 발생한 경우 -1을 반환합니다(데이터가 아직 계산되지 않음).

© 2000-2025, MetaQuotes Ltd.

GetData

지표의 지정된 버퍼에서 지정된 요소를 가져옵니다. Refresh()를 메소드를 사용하기 전에 최근 데이터로 작업하기 위해 호출해야 합니다.

```
double  GetData(
   const int  buffer_num,     // 버퍼명
   const int  index           // 색인
   ) const
```

Parameters

buffer_num

[in] 지표 버퍼 수.

index

[in] 지표 버퍼 요소 색인.

Return Value

value – 성공, EMPTY_VALUE – 데이터를 받을 수 없음.

GetData

시작 위치 및 번호별로 지표의 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   const int      start_pos,      // 위치
   const int      count,          // 숫자
   const int      buffer_num,     // 버퍼 수
   double&        buffer[]        // 배열
   ) const
```

Parameters

start_pos

[in] 지표 버퍼의 시작 위치.

count

[in] 지표 버퍼 요소 수.

buffer_num

[in] 지표 버퍼 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

지정된 지표 버퍼에서 받은 지표 값 수 – 성공, 그렇지 않으면 -1입니다.

GetData

지표 버퍼에서 시작 시간 및 번호별로 데이터를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   const datetime  start_time,     // 시작 시간
   const int       count,          // 양
   const int       buffer_num,     // 버퍼 수
   double&         buffer[]        // 배열
   ) const
```

Parameters

start_time

[in] 지표 버퍼 요소 시작 시간.

count

[in] 지표 버퍼 요소 수.

buffer_num

[in] 지표 버퍼 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

지정된 버퍼에서 수신한 지표 값 수, 그렇지 않으면 -1.

GetData

시작 및 종료 시간별로 지표 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   const datetime  start_time,     // 시작 시간
   const datetime  stop_time,      // 종료 시간
   const int       buffer_num,     // 버퍼 수
   double&         buffer[]        // 배열
   ) const
```

Parameters

start_time

[in] 지표 버퍼 초기 요소 시간.

stop_time

[in] 지표 버퍼 끝 요소 시간.

buffer_num

[in] 지표 버퍼 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

지정된 버퍼에서 받은 지표 값 수 - 성공, 그렇지 않으면 -1.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Refresh

지표 데이터를 업데이트 합니다. GetData()를 사용하기 전에 메소드를 호출하는 것이 좋습니다.

```
virtual void  Refresh(
   int  flags=OBJ_ALL_PERIODS      // 플래그
   )
```

Parameters

flags=OBJ_ALL_PERIODS

[in] 타임프레임 업데이트 플래그.

© 2000-2025, MetaQuotes Ltd.

Minimum

지정한 범위에서 지정한 버퍼의 최소 요소 인덱스를 반환합니다.

```
int  Minimum(
   const int  buffer_num,     // 버퍼명
   const int  start,          // 시작 인덱스
   const int  count           // 수
   ) const
```

Parameters

buffer_num

[in] 값을 검색할 버퍼 번호.

start

[in] 검색 범위 초기 인덱스.

count

[in] 검색 범위 크기(요소 수).

Return Value

지정된 범위에 있는 지정된 버퍼의 최소 요소 인덱스.

© 2000-2025, MetaQuotes Ltd.

MinValue

지정한 범위에서 지정한 버퍼의 최소 요소 값을 반환합니다.

```
double  MinValue(
   const int   buffer_num,     // 버퍼 수
   const int   start,          // 시작 인덱스
   const int   count,          // 수
   int&        index           // 참조
   ) const
```

Parameters

buffer_num

[in] 값을 검색할 버퍼 번호.

start

[in] 검색 범위 초기 인덱스.

count

[in] 검색 범위 크기(요소 수).

index

[out] 찾은 요소 인덱스 값을 저장하기 위한 변수에 대한 참조.

Return Value

지정한 범위에 있는 지정된 버퍼의 최소 요소 값.

Note

찾은 요소의 인덱스는 인덱스 참조에 의해 저장됩니다.

© 2000-2025, MetaQuotes Ltd.

Maximum

지정한 범위에서 지정한 버퍼의 최대 요소 인덱스를 반환합니다.

```
int  Maximum(
   const int  buffer_num,     // 버퍼명
   const int  start,          // 시작 인덱스
   const int  count           // 수
   ) const
```

Parameters

buffer_num

[in] 값을 검색할 버퍼 번호.

start

[in] 검색 범위 초기 인덱스.

count

[in] 검색 범위 크기(요소 수).

Return Value

지정된 범위에 있는 지정된 버퍼의 최대 요소 인덱스.

© 2000-2025, MetaQuotes Ltd.

MaxValue

지정한 범위에서 지정한 버퍼의 최대 요소 값을 반환합니다.

```
double  MaxValue(
   const int   buffer_num,     // 버퍼 수
   const int   start,          // 시작 인덱스
   const int   count,          // 수
   int&        index           // 참조
   ) const
```

Parameters

buffer_num

[in] 값을 검색할 버퍼 번호.

start

[in] 검색 범위 초기 인덱스.

count

[in] 검색 범위 크기(요소 수).

index

[out] 찾은 요소 인덱스 값을 저장하기 위한 변수에 대한 참조.

Return Value

지정된 범위에 있는 지정된 버퍼의 최대 요소 값.

Note

최대 버퍼 요소의 인덱스는 인덱스 참조에 의해 저장됩니다.

© 2000-2025, MetaQuotes Ltd.

MethodDescription

ENUM_MA_METHOD열거값을 문자열로 전환합니다.

```
string  MethodDescription(
   const int  val      // 값
   ) const
```

Parameters

val

[in] 전환 값.

Return Value

ENUM_MA_METHOD 열거 값에 상응하는 문자열.

© 2000-2025, MetaQuotes Ltd.

PriceDescription

ENUM_APPLIED_PRICE 열거 값을 문자열로 전환합니다.

```
string  PriceDescription(
   const int  val      // 값
   ) const
```

Parameters

val

[in] 전환 값.

Return Value

ENUM_APPLIED_PRICE 열거 값에 상응하는 문자열.

© 2000-2025, MetaQuotes Ltd.

VolumeDescription

ENUM_APPLIED_VOLUME 열거 값을 문자열로 전환합니다.

```
string  VolumeDescription(
   const int  val      // 값
   ) const
```

Parameters

val

[in] 전환 값.

Return Value

ENUM_APPLIED_VOLUME 열거 값에 상응하는 문자열.

© 2000-2025, MetaQuotes Ltd.

AddToChart

차트에 지표를 추가합니다.

```
bool  AddToChart(
   const long  chart,      // 차트 ID
   const int  subwin       // 하위창 색인
   )
```

Parameters

chart

[in] 차트 ID.

subwin

[in] 차트 하위창 색인.

Return Value

true – 성공, false – 차트에 지표를 추가할 수 없음.

© 2000-2025, MetaQuotes Ltd.

DeleteFromChart

차트에서 지표를 삭제합니다.

```
bool  DeleteFromChart(
   const long  chart,      // 차트 ID
   const int  subwin       // 하위창 색인
   )
```

Parameters

chart

[in] 차트 ID.

subwin

[in] 차트 하위창 색인.

Return Value

true – 성공, false – 차트에서 지표를 제거할 수 없음.

© 2000-2025, MetaQuotes Ltd.

CIndicators

CIndicators는 시계열 및 기술 지표 클래스의 인스턴스를 수집하기 위한 클래스입니다.

Description

CIndicators 클래스는 기술 지표 클래스 인스턴스 생성, 저장 및 관리(데이터 동기화, 핸들 및 메모리 관리)를 제공합니다.

Declaration

```
class CIndicators: public CArrayObj
```

Title

```
#include <Indicators\Indicators.mqh>
```

그룹별 클래스 메서드

Create	
Create	지표를 생성합니다
데이타 업데이트	
Refresh	지표 데이터 업데이트

© 2000-2025, MetaQuotes Ltd.

Create

지정한 매개 변수를 사용하여 지정된 유형의 지표를 작성합니다.

```
CIndicator*  Create(
   const string           symbol,     // 심볼명
   const ENUM_TIMEFRAMES  period,     // 주기
   const ENUM_INDICATOR   type,       // 유형
   const int              count,      // 매개변수의 수
   const MqlParam&        params      // 매개변수 배열
   )
```

Parameters

symbol

[in] 지표 심볼명.

period

[in] 지표 타임프레임(ENUM_TIMEFRAMES 열거 값).

type

[in] 지표 유형(ENUM_INDICATOR 열거 값).

count

[in] 지표의 매개변수의 수.

params

[in] 지표의 매개변수 배열 참조.

Return Value

생성된 지표의 참조 - 성공, NULL - 지표를 생성할 수 없음.

© 2000-2025, MetaQuotes Ltd.

Refresh

컬렉션의 모든 시계열 및 기술 지표에 대한 데이터를 업데이트합니다.

```
int  Refresh()
```

Return Value

타임프레임 플래그가 업데이트됨(개체 가시성 플래그로 형성됨).

© 2000-2025, MetaQuotes Ltd.

Timeseries classes

이 챕터 그룹에는 MQL5 표준 라이브러리의 시계열 클래스에 대한 기술적 세부 정보와 모든 주요 구성 요소에 대한 설명이 포함되어 있습니다.

Class	Description
CiSpread	과거 데이터 확산에 대한 액세스를 제공합니다
CiTime	과거 막대의 빈 시간에 대한 액세스를 제공합니다
CiTickVolume	과거 막대의 틱 볼륨에 대한 액세스를 제공합니다
CiRealVolume	과거 막대의 실제 볼륨에 대한 액세스를 제공합니다
CiOpen	과거 막대의 시가에 대한 액세스를 제공합니다
CiHigh	과거 막대의 고가에 대한 액세스를 제공합니다
CiLow	과거 막대의 저가에 대한 액세스를 제공합니다
CiClose	과거 막대의 종가에 대한 액세스를 제공합니다

© 2000-2025, MetaQuotes Ltd.

CiSpread

CiSpread는 과거 막대의 스프레드에 액세스할 수 있도록 설계된 클래스입니다.

Description

CiSpread 클래스는 과거 데이터를 확산하기 위한 액세스를 제공합니다.

Declaration

```
class CiSpread: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CiSpread

그룹별 클래스 메서드

Create	
Create	시계열을 생성합니다
BufferResize	계열 버퍼 크기를 설정합니다
데이터 액세스	
GetData	계열 데이터를 가져옵니다
데이타 업데이트	
Refresh	계열 데이터 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

스프레드 히스토리에 액세스를 위해 지정된 매개 변수와 시계열을 만듭니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 시계열 심볼.

period

[in] 시계열 타임프레임(ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 시계열 생성 불가.

© 2000-2025, MetaQuotes Ltd.

BufferResize

버퍼 시리즈의 크기를 설정합니다.

```
virtual void  BufferResize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

GetData

시계열 버퍼의 지정된 요소를 가져옵니다.

```
int  GetData(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 인덱스.

Return Value

시계열 버퍼 요소, 또는 0.

GetData

시작 위치 및 요소 수별로 시계열의 요소를 가져옵니다.

```
int  GetData(
   int   start_pos,     // 위치
   int   count,         // 숫자
   int&  buffer         // 배열
   ) const
```

Parameters

start_pos

[in] 시계열 버퍼의 시작 위치.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

성공시 >=0, 데이터를 받을 수 없을 경우 -1.

GetData

초기 시간 및 번호로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   int       count,          // 수
   int&      buffer          // 배열
   ) const
```

Parameters

start_time

© 2000-2025, MetaQuotes Ltd.

[in] 시계열 버퍼의 초기 요소 시간.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 및 정지 시간별로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   datetime  stop_time,      // 정지 시간
   int&      buffer          // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 요소의 시작 시간.

stop_time

[in] 시계열 버퍼 요소의 중지 시간.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

성공시 >=0, 데이터를 받을 수 없을 경우 -1.

© 2000-2025, MetaQuotes Ltd.

Refresh

시계열 데이터 업데이트.

```
virtual void  Refresh(
   int  flags      // 플래그
   )
```

Parameters

flags

[in] 타임프레임 플래그.

© 2000-2025, MetaQuotes Ltd.

CiTime

CiTime은 과거 술집 오픈 시간에 액세스할 수 있도록 설계된 클래스입니다.

Description

CiTime 클래스는 과거의 막대의 빈 시간에 대한 액세스를 제공합니다.

Declaration

```
class CiTime: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CiTime

그룹별 클래스 메서드

Create	
Create	시계열을 생성합니다
BufferResize	시계열 버퍼 크기를 설정합니다
데이터 액세스	
GetData	시계열 데이터를 가져옵니다
데이타 업데이트	
Refresh	시계열 데이터 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

기록에 있는 막대의 시작 시간에 액세스하기 위해 지정된 매개 변수를 사용하여 시계열을 만듭니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 시계열 심볼.

period

[in] 시계열 타임프레임(ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 시계열 생성 불가.

© 2000-2025, MetaQuotes Ltd.

BufferResize

계열 버퍼 크기를 설정합니다.

```
virtual void  BufferResize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

GetData

시계열 버퍼의 지정된 요소를 가져옵니다.

```
datetime  GetData(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 인덱스.

Return Value

시계열 버퍼 요소, 또는 0.

GetData

시작 위치 및 번호별로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   int   start_pos,     // 위치
   int   count,         // 숫자
   long& buffer         // 배열
   ) const
```

Parameters

start_pos

[in] 시계열의 시작 포지션.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 시간 및 번호로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   int       count,          // 수
   long&     buffer          // 배열
   ) const
```

Parameters

start_time

© 2000-2025, MetaQuotes Ltd.

[in] 시계열 버퍼 초기 요소의 시간.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 및 중지 시간을 기준으로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   datetime  stop_time,      // 정지 시간
   long&     buffer          // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

stop_time

[in] 시계열 버퍼 종결 요소 시간.

buffer

[in] 데이터를 저장하기 위한 배열 참조

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

© 2000-2025, MetaQuotes Ltd.

Refresh

시계열 데이터 업데이트.

```
virtual void  Refresh(
   int  flags      // 플래그
   )
```

Parameters

flags

[in] 타임프레임 플래그.

© 2000-2025, MetaQuotes Ltd.

CiTickVolume

CiTickVolume은 과거 막대의 틱 볼륨에 액세스하기 위해 설계된 클래스입니다.

Description

CiTickVolume 클래스는 과거 막대의 틱 볼륨에 대한 액세스를 제공합니다.

Declaration

```
class CiTickVolume: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CiTickVolume

그룹별 클래스 메서드

Create	
Create	시계열을 생성합니다
BufferResize	버퍼 크기 설정
데이터 액세스	
GetData	계열 데이터를 가져옵니다
데이타 업데이트	
Refresh	계열 데이터 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

과거 막대의 틱 볼륨에 액세스하기 위해 지정된 매개변수를 사용하여 시계열을 작성합니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 시계열 심볼.

period

[in] 시계열 타임프레임(ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 시계열 생성 불가.

© 2000-2025, MetaQuotes Ltd.

BufferResize

계열 버퍼 크기를 설정합니다.

```
virtual void  BufferResize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

GetData

지정된 시계열 버퍼 요소를 가져옵니다.

```
long  GetData(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소의 인덱스.

Return Value

시계열 버퍼 요소, 또는 0.

GetData

시작 위치 및 번호별로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   int   start_pos,     // 위치
   int   count,         // 숫자
   long& buffer         // 배열
   ) const
```

Parameters

start_pos

[in] 시계열 버퍼의 시작 위치.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 시간 및 번호로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   int       count,          // 수
   long&     buffer          // 배열
   ) const
```

Parameters

start_time

© 2000-2025, MetaQuotes Ltd.

[in] 시계열 버퍼 초기 요소 시간.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터를 저장하기 위한 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 및 중지 시간을 기준으로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   datetime  stop_time,      // 정지 시간
   long&     buffer          // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소 시간.

stop_time

[in] 시계열 버퍼 끝 요소의 시간.

buffer

[in] 데이터를 저장하기 위한 배열 참조

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

© 2000-2025, MetaQuotes Ltd.

Refresh

시계열 데이터 업데이트.

```
virtual void  Refresh(
   int  flags      // 플래그
   )
```

Parameters

flags

[in] 타임프레임 플래그.

© 2000-2025, MetaQuotes Ltd.

CiRealVolume

CiRealVolumeは 과거 막대의 실제 볼륨에 액세스할 수 있도록 설계된 클래스입니다.

Description

CiRealVolume 클래스는 과거 막대의 실제 볼륨에 대한 액세스를 제공합니다.

Declaration

```
class CiRealVolume: public CSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
  CArray
    CArrayObj
      CSeries
        CiRealVolume

그룹별 클래스 메서드

Create	
Create	계열 생성
BufferResize	버퍼 사이즈 설정
데이터 액세스	
GetData	계열 데이터를 가져옵니다
데이타 업데이트	
Refresh	계열 데이터 업데이트

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

Create

과거 막대의 실제 볼륨에 액세스하기 위해 지정된 매개변수로 시계열을 작성합니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 시계열 심볼.

period

[in] 시계열 타임프레임(ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 시계열 생성 불가.

© 2000-2025, MetaQuotes Ltd.

BufferResize

계열 버퍼 크기 설정

```
virtual void  BufferResize(
   int  size      // 사이즈
   )
```

Parameters

size

[in] 새 버퍼 크기.

© 2000-2025, MetaQuotes Ltd.

GetData

지정된 시계열 버퍼 요소를 가져옵니다.

```
datetime  GetData(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

계열 버퍼 요소, 또는 0.

GetData

시작 위치 및 번호별로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   int   start_pos,     // 위치
   int   count,         // 숫자
   long& buffer         // 배열
   ) const
```

Parameters

start_pos

[in] 시계열 버퍼의 시작 위치.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 시간 및 번호로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   int       count,          // 수
   long&     buffer          // 배열
   ) const
```

Parameters

start_time

© 2000-2025, MetaQuotes Ltd.

[in] 시계열 버퍼 초기 요소 시간.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 및 중지 시간을 기준으로 시계열 요소를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   datetime  stop_time,      // 정지 시간
   long&     buffer          // 목표 배열
   ) const
```

Parameters

start_time

[in] 시작 시간.

stop_time

[in] 정지 시간.

buffer

[in] 데이터의 대상 배열에 대한 참조

Return Value

성공적이면 >=0, 오류일 경우 -1.

© 2000-2025, MetaQuotes Ltd.

Refresh

시계열 데이터 업데이트.

```
virtual void  Refresh(
   int  flags      // 플래그
   )
```

Parameters

flags

[in] 타임프레임 플래그.

© 2000-2025, MetaQuotes Ltd.

CiOpen

CiOpen은 과거 막대의 시가에 접근할 수 있도록 설계된 클래스입니다.

Description

CiOpen 클래스는 과거 막대의 시가에 대한 액세스를 제공합니다.

Declaration

```
class CiOpen: public CPriceSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiOpen

그룹별 클래스 메서드

Create	
Create	계열 생성
데이터 액세스	
GetData	계열 데이터를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CPriceSeries 클래스에서 상속된 메서드

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

과거 막대의 시가에 액세스하기 위해 지정된 매개 변수를 사용하여 시계열을 작성합니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 시계열 심볼.

period

[in] 시계열 타임프레임(ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 시계열 생성 불가.

© 2000-2025, MetaQuotes Ltd.

GetData

시작 위치 및 번호별로 시계열 요소를 가져옵니다.

```
int  GetData(
   int    start_pos,      // 시작 포지션
   int    count,          // 숫자
   double& buffer         // 배열
   ) const
```

Parameters

start_pos

[in] 시계열 버퍼의 시작 위치.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 시간 및 번호로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,      // 시작 시간
   int       count,           // 수
   double&     buffer         // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 및 중지 시간을 기준으로 시계열 버퍼에서 데이터를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   datetime  stop_time,      // 정지 시간
   double&   buffer          // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

stop_time

[in] 시계열 버퍼 마지막 요소의 시간.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

© 2000-2025, MetaQuotes Ltd.

CiHigh

CiHigh는 과거 막대 중 고가에 액세스할 수 있도록 설계된 클래스입니다.

Description

CiHigh 클래스는 과거 막대 중 고가에 액세스할 수 있게 해줍니다.

Declaration

```
class CiHigh: public CPriceSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiHigh

그룹별 클래스 메서드

Create	
Create	계열 생성
데이터 액세스	
GetData	계열 데이터를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CPriceSeries 클래스에서 상속된 메서드

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

과거 막대의 고가에 액세스하기 위해 지정된 매개 변수를 사용하여 시계열을 작성합니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 시계열 심볼.

period

[in] 시계열 타임프레임(ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 시계열 생성 불가.

© 2000-2025, MetaQuotes Ltd.

GetData

시작 위치 및 번호별로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   int    start_pos,      // 위치
   int    count,          // 숫자
   double& buffer         // 배열
   ) const
```

Parameters

start_pos

[in] 시계열 버퍼의 시작 위치.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 시간 및 번호로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,      // 시작 시간
   int       count,           // 수
   double&     buffer         // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 및 중지 시간을 기준으로 시계열 버퍼에서 데이터를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   datetime  stop_time,      // 정지 시간
   double&   buffer          // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

stop_time

[in] 시계열 버퍼 마지막 요소의 시간.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

© 2000-2025, MetaQuotes Ltd.

CiLow

CiLow는 과거 저가 막대에 액세스 할 수 있도록 설계된 클래스입니다.

Description

CiLow 클래스는 과거 저가 막대에 액세스할 수 있는 기회를 제공합니다.

Declaration

```
class CiLow: public CPriceSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiLow

그룹별 클래스 메서드

Create	
Create	계열 생성
데이터 액세스	
GetData	계열 데이터를 얻습니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CPriceSeries 클래스에서 상속된 메서드

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

과거에서 막대의 저가에 액세스하기 위해 지정된 매개 변수를 사용하여 시계열을 작성합니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 시계열 심볼.

period

[in] 시계열 타임프레임(ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 시계열 생성 불가.

© 2000-2025, MetaQuotes Ltd.

GetData

시작 위치 및 요소 수별로 시계열의 요소를 가져옵니다.

```
int  GetData(
   int    start_pos,      // 위치
   int    count,          // 숫자
   double& buffer         // 배열
   ) const
```

Parameters

start_pos

[in] 시계열 버퍼의 시작 위치.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 시간 및 번호로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   int       count,          // 수
   double&     buffer        // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 및 중지 시간을 기준으로 시계열 버퍼에서 데이터를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   datetime  stop_time,      // 정지 시간
   double&   buffer          // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

stop_time

[in] 시계열 버퍼 마지막 요소의 시간.

buffer

[in] 데이터 저장소 배열 참조

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

© 2000-2025, MetaQuotes Ltd.

CiClose

CiClose는 과거 막대의 종가에 접근할 수 있도록 설계된 클래스입니다.

Description

CiClose 클래스는 과거 막대의 종가에 대한 액세스를 제공합니다.

Declaration

```
class CiClose: public CPriceSeries
```

Title

```
#include <Indicators\TimeSeries.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiClose

그룹별 클래스 메서드

Create	
Create	계열 생성
데이터 액세스	
GetData	계열 데이터 가져오기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Type, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CPriceSeries 클래스에서 상속된 메서드

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

© 2000-2025, MetaQuotes Ltd.

Create

과거 막대의 종가에 액세스하기 위해 지정된 매개 변수를 사용하여 시계열을 작성합니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 시계열 심볼.

period

[in] 시계열 타임프레임(ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 시계열 생성 불가.

© 2000-2025, MetaQuotes Ltd.

GetData

시작 위치 및 번호별로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   int    start_pos,      // 위치
   int    count,          // 숫자
   double& buffer         // 배열
   ) const
```

Parameters

start_pos

[in] 시계열 버퍼의 시작 위치.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 시간 및 번호로 시계열 버퍼에서 데이터를 가져옵니다.

```
int  GetData(
   datetime  start_time,      // 시작 시간
   int       count,           // 수
   double&     buffer         // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

count

[in] 시계열 버퍼 요소 수.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

GetData

시작 및 중지 시간을 기준으로 시계열 버퍼에서 데이터를 가져옵니다.

© 2000-2025, MetaQuotes Ltd.

```
int  GetData(
   datetime  start_time,     // 시작 시간
   datetime  stop_time,      // 정지 시간
   double&   buffer          // 배열
   ) const
```

Parameters

start_time

[in] 시계열 버퍼 초기 요소의 시간.

stop_time

[in] 시계열 버퍼 마지막 요소의 시간.

buffer

[in] 데이터 저장소 배열 참조.

Return Value

>=0 – 성공, -1 – 데이터를 받을 수 없음.

© 2000-2025, MetaQuotes Ltd.

기술 지표 및 시계열의 주요 및 보조 클래스

이 챕터 그룹에는 MQL5 표준 라이브러리의 적절한 구성 요소에 대한 설명뿐만 아니라 기술 지표 및 시계열의 주요 및 보조 클래스에 대한 기술적 세부 정보가 포함되어 있습니다.

클래스/그룹	Description
CiADX	평균 지향성 색인
CiADXWilder	웰즈 와일더별 평균 방향지수
CiBands	Bollinger Bands®
CiEnvelopes	Envelopes
CiIchimoku	일목균형표
CiMA	이동 평균
CiSAR	파라볼릭 SAR 시스템
CiStdDev	표준 편차
CiDEMA	이중 지수 이동 평균
CiTEMA	삼중 지수 이동 평균
CiFrAMA	프랙탈 적응형 이동 평균
CiAMA	적응형 이동 평균
CiVIDyA	가변 인덱스 동적 평균

© 2000-2025, MetaQuotes Ltd.

CiADX

CiADX는 평균 방향 지수 기술 지표를 사용하기 위한 클래스입니다.

Description

CiADX 클래스는 평균 방향 인덱스 지표의 작성, 구성 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiADX: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

COBject
CArray
CArrayObj
CSeries
CIndicator
CiADX

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	메인 선의 버퍼 요소를 반환
Plus	+DI 라인의 버퍼 요소를 반환
Minus	-DI 선의 버퍼 요소를 반환
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균화 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

메인 선의 버퍼 요소를 지정한 인덱스별로 반환

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 메인 선 버퍼 요소 색인.

Return Value

지정된 인덱스의 주선 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Plus

지정한 인덱스별로 +DI 줄의 버퍼 요소를 반환합니다.

```
double  Plus(
   int  index      // 인덱스
   )
```

Parameters

index

[in] +DI 라인 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 +DI 줄의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Minus

지정한 인덱스별로 -DI 줄의 버퍼 요소를 반환합니다.

```
double  Minus(
   int  index      // 인덱스
   )
```

Parameters

index

[in] -DI 라인 버퍼 요소 인덱스.

Return Value

지정된 인덱스에 의한 -DI 줄의 버퍼 요소, 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiADX용 IND_ADX).

© 2000-2025, MetaQuotes Ltd.

CiADXWilder

CiADXWilder는 웰스 와일더의 기술 지표 평균 방향 지수를 사용하기 위한 클래스입니다.

Description

CiADXWilder 클래스는 웰스 와일더의 평균 방향 지수의 데이터에 대한 작성, 구성 및 액세스를 제공합니다.

Declaration

```
class CiADXWilder: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADXWilder

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	메인 선의 버퍼 데이터를 반환합니다
Plus	+DI 라인의 버퍼 데이터를 반환합니다
Minus	-DI 선의 버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균화 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

메인 선의 버퍼 요소를 지정한 인덱스별로 반환

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 메인 선 버퍼 요소 색인.

Return Value

지정된 인덱스의 주선 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Plus

지정한 인덱스별로 +DI 줄의 버퍼 요소를 반환합니다.

```
double  Plus(
   int  index      // 인덱스
   )
```

Parameters

index

[in] +DI 라인 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 +DI 줄의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Minus

지정한 인덱스별로 -DI 줄의 버퍼 요소를 반환합니다.

```
double  Minus(
   int  index      // 인덱스
   )
```

Parameters

index

[in] -DI 라인 버퍼 요소 인덱스.

Return Value

지정된 인덱스에 의한 -DI 라인의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiADXWilder용 IND_ADXW).

© 2000-2025, MetaQuotes Ltd.

CiBands

CiBands는 Bollinger Bands® 기술 지표를 사용하기 위한 클래스입니다.

Description

CiBands 클래스는 볼링거 밴드 표시기의 데이터에 대한 작성, 구성 및 액세스를 제공합니다.

Declaration

```
class CiBands: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBands

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
MaShift	수평 이동을 반환합니다
Deviation	편차를 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Base	기준선의 버퍼 요소를 반환합니다
Upper	상한선의 버퍼 요소를 반환합니다
Lower	하한선의 버퍼 요소를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaShift

수평 이동을 반환합니다.

```
int  MaShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Deviation

편차를 반환합니다.

```
double  Deviation() const
```

Return Value

지표 작성 시 정의된 편차를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              ma_shift,      // 시프트
   double           deviation,     // 편차
   int              applied        // 적용 가격, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

ma_shift

[in] 지표의 수평 이동.

deviation

[in] 편차.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Base

기준선의 버퍼 요소를 지정한 인덱스별로 반환합니다.

```
double  Base(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 기준선 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 기준선의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Upper

지정한 인덱스별로 위쪽 줄의 버퍼 요소를 반환합니다.

```
double  Upper(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 상한선 버퍼 요소 색인.

Return Value

지정된 인덱스의 위쪽 줄에 있는 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Lower

지정한 인덱스별로 하한선의 버퍼 요소를 반환합니다.

```
double  Lower(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 하한선 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 아래쪽 줄의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE .

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiBands용 IND_BANDS).

© 2000-2025, MetaQuotes Ltd.

CiEnvelopes

CiEnvelopes는 Envelops 기술 지표를 사용하기 위한 클래스입니다.

Description

CiEnvelopes 클래스는 Envelops 지표의 생성, 구성 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiEnvelopes: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiEnvelopes

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
MaShift	수평 이동을 반환합니다
MaMethod	평균화 방법을 반환합니다
Deviation	편차를 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Upper	상한선의 버퍼 데이터를 반환합니다.
Lower	하한선의 버퍼 데이터를 반환합니다.
Input/output	
virtual Type	가삭 식별 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaShift

지표의 수평 이동을 반환합니다.

```
int  MaShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaMethod

평균화 메서드를 반환합니다

```
ENUM_MA_METHOD  MaMethod() const
```

Return Value

지표 작성 시 정의된 평균화 방법을 반환합니다(ENUM_MA_METHOD 열거값).

© 2000-2025, MetaQuotes Ltd.

Deviation

편차 값을 반환합니다.

```
double  Deviation() const
```

Return Value

지표 작성 시 정의된 편차 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              ma_shift,      // 시프트
   ENUM_MA_METHOD   ma_method,     // 평균화 메서드
   int              applied,       // 가격 유형, 핸들
   double           deviation      // 편차
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

ma_shift

[in] 가격 축 이동.

ma_method

[in] 평균화 메서드 (ENUM_MA_METHOD 열거값).

applied

[in] 적용할 객체(가격 유형 또는 핸들).

deviation

[in] 편차.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Upper

지정한 인덱스별로 위쪽 줄의 버퍼 요소를 반환합니다.

```
double  Upper(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 상한선 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 위쪽 줄의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Lower

지정한 인덱스별로 하한선의 버퍼 요소를 반환합니다.

```
double  Lower(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 하한선 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 아래쪽 줄의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE .

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiEnvelopes용 IND_ENVELOPES).

© 2000-2025, MetaQuotes Ltd.

CiIchimoku

CiIchimoku는 이치모쿠 긴코 효 기술 지표를 사용하기 위한 클래스입니다.

Description

CiIchimoku 클래스는 이치모쿠 긴코 효 지표의 생성, 설정 및 데이터 액세스를 제공합니다.

Declaration

```
class CiIchimoku: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiIchimoku

그룹별 클래스 메서드

Attributes	
TenkanSenPeriod	TenkanSen 기간을 반환합니다
KijunSenPeriod	KijunSen 기간을 반환합니다
SenkouSpanBPeriod	SenkouSpanB 기간을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
TenkanSen	TenkanSen 선의 버퍼 요소를 반환합니다
KijunSen	KijunSen 선의 버퍼 요소를 반환합니다
SenkouSpanA	SenkouSpanA 라인의 버퍼 요소를 반환합니다
SenkouSpanB	SenkouSpanB 라인의 버퍼 요소를 반환합니다
ChinkouSpan	ChikouSpan 선의 버퍼 요소를 반환합니다
Input/output	

© 2000-2025, MetaQuotes Ltd.

Attributes	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

TenkanSenPeriod

TenkanSen 기간을 반환합니다.

```
int  TenkanSenPeriod() const
```

Return Value

지표 작성 시 정의된 TenkanSen 기간을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

KijunSenPeriod

KijunSen 기간을 반환합니다.

```
int  KijunSenPeriod() const
```

Return Value

지표 작성시KijunSen 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanBPeriod

SenkouSpanB 주기를 반환합니다.

```
int  SenkouSpanBPeriod() const
```

Return Value

지표 작성시 정의된 SenkouSpanB 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,           // 심볼
   ENUM_TIMEFRAMES  period,           // 주기
   int              tenkan_sen,       // TenkanSen 주기
   int              kijun_sen,        // KijunSen 주기
   int              senkou_span_b     // SenkouSpanB 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

tenkan_sen

[in] TenkanSen 주기.

kijun_sen

[in] KijunSen 주기.

senkou_span_b

[in] SenkouSpanB 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

TenkanSen

지정한 인덱스별로 TenkanSen 선의 버퍼 요소를 반환합니다.

```
double  TenkanSen(
   int  index      // 인덱스
   )
```

Parameters

index

[in] TenkanSen 선 버퍼 요소 색인.

Return Value

지정된 인덱스별 TenkanSen 선의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

KijunSen

지정된 인덱스별로 KijunSen 선의 버퍼 요소를 반환합니다.

```
double  KijunSen(
   int  index      // 인덱스
   )
```

Parameters

index

[in] KijunSen 라인 버퍼 요소 색인.

Return Value

지정된 인덱스의 KijunSen 줄의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanA

지정한 인덱스별로 SenkouSpanA 라인의 버퍼 요소를 반환합니다.

```
double  SenkouSpanA(
   int  index      // 인덱스
   )
```

Parameters

index

[in] SenkouSpanA 라인 버퍼 요소 지수.

Return Value

지정된 인덱스의 SenkouSpanA 줄에 있는 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

SenkouSpanB

지정한 인덱스별로 SenkouSpanB 선의 버퍼 요소를 반환합니다.

```
double  SenkouSpanB(
   int  index      // 인덱스
   )
```

Parameters

index

[in] SenkouSpanB 라인 버퍼 요소 지수.

Return Value

지정된 인덱스의 SenkouSpanB 라인에 있는 버퍼 요소 또는 올바른 데이터가 없는 경우EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

ChinkouSpan

지정한 인덱스별로 ChinkouSpan 선의 버퍼 요소를 반환합니다.

```
double  ChinkouSpan(
   int  index      // 인덱스
   )
```

Parameters

index

[in] ChinkouSpan 선 버퍼 요소 지수.

Return Value

지정된 인덱스의 ChinkouSpan 선의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(Cilchimoku용 IND_ICHIMOKU).

© 2000-2025, MetaQuotes Ltd.

CiMA

CiMA는 이동 평균 기술 지표를 사용하기 위한 클래스입니다.

Description

CiMA 클래스는 이동 평균 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMA

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
MaShift	수평 이동을 반환합니다
MaMethod	평균화 방법을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaShift

지표의 수평 이동을 반환합니다.

```
int  MaShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaMethod

평균화 메서드를 반환합니다

```
ENUM_MA_METHOD  MaMethod() const
```

Return Value

지표 작성시 정의된 평균화 메서드(ENUM_MA_METHOD 열거 값)를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           string,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              ma_shift,      // 시프트
   ENUM_MA_METHOD   ma_method,     // 평균화 메서드
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

string

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

ma_shift

[in] Horizontal shift.

ma_method

[in] 평균화 메서드 (ENUM_MA_METHOD 열거값).

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiMA용 IND_MA).

© 2000-2025, MetaQuotes Ltd.

CiSAR

CiSAR은 파라볼릭 SAR 시스템 기술 지표를 사용하기 위한 클래스입니다.

Description

CiSAR 클래스는 파라볼릭 SAR 시스템 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiSAR: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiSAR

그룹별 클래스 메서드

Attributes	
SarStep	가격 증가 단계를 반환합니다
Maximum	스텝의 최대값을 반환합니다
메서드 생성	
Create	지표를 작성합니다
데이터 액세스 메서드	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

SarStep

가격 증가 단계를 반환합니다.

```
double  SarStep() const
```

Return Value

지표 작성 시 정의된 가격 상승 단계입니다.

© 2000-2025, MetaQuotes Ltd.

Maximum

스텝의 최대값을 반환합니다.

```
double  Maximum() const
```

Return Value

지표 작성시 정의된 단계의 최대 값.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,      // 심볼
   ENUM_TIMEFRAMES  period,      // 주기
   double           step,        // 스텝
   double           maximum      // 계수
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

step

[in] 속도 증가를 위한 단계.

maximum

[in] 계수에 따른 가격.

Return Value

true – 성공, false – 표시를 바꿀 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiSAR용 IND_SAR).

© 2000-2025, MetaQuotes Ltd.

CiStdDev

CiStdDev는 표준 편차 기술 지표를 사용하기 위한 클래스입니다.

Description

CiStdDev 클래스는 표준 편차 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiStdDev: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStdDev

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
MaShift	수평 이동을 반환합니다
MaMethod	평균화 방법을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaShift

지표의 수평 이동을 반환합니다.

```
int  MaShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaMethod

평균화 메서드를 반환합니다

```
ENUM_MA_METHOD  MaMethod() const
```

Return Value

지표 작성시 정의된 평균화 메서드(ENUM_MA_METHOD 열거값)을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              ma_shift,      // 시프트
   ENUM_MA_METHOD   ma_method,     // 평균화 메서드
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

ma_shift

[in] 수평 시프트.

ma_method

[in] 평균화 메서드 (ENUM_MA_METHOD 열거값).

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

성공한 경우 지정된 인덱스를 기준으로 버퍼 요소, 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiStdDev용 IND_STDDEV).

© 2000-2025, MetaQuotes Ltd.

CiDEMA

CiDEMA는 이중 지수 이동 평균 기술 지표를 사용하기 위한 클래스입니다.

Description

CiDEMA 클래스는 이중 지수 이동 평균 표시기의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiDEMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiDEMA

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
IndShift	수평 이동을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

IndShift

지표의 수평 이동을 반환합니다.

```
int  IndShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           string,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              ind_shift,     // 시프트
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

string

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

ind_shift

[in] 수평 시프트.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiDEMA용 IND_DEMA).

© 2000-2025, MetaQuotes Ltd.

CiTEMA

CiTEMA는 삼중 지수 이동 평균 기술 지표를 사용하기 위한 클래스입니다.

Description

CiTEMA 클래스는 삼중 지수 이동 평균 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiTEMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTEMA

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
IndShift	수평 이동을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

IndShift

지표의 수평 이동을 반환합니다.

```
int  IndShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              ma_shift,      // 시프트
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

ma_shift

[in] 수평 시프트.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiTEMA용 IND_TEMA).

© 2000-2025, MetaQuotes Ltd.

CiFrAMA

CiFramA는 프랙탈 적응형 이동 평균 기술 지표를 사용하기 위한 클래스입니다.

Description

CiFrAMA 클래스는 프랙탈 적응형 이동 평균 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiFrAMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFrAMA

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
IndShift	수평 이동을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

IndShift

지표의 수평 이동을 반환합니다.

```
int  IndShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              ma_shift,      // 시프트
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

ma_shift

[in] 수평 시프트.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스의 버퍼 요소(성공한 경우) 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiFrAMA용 IND_FRAMA).

© 2000-2025, MetaQuotes Ltd.

CiAMA

CiAMA는 적응형 이동 평균 기술 지표를 사용하기 위한 클래스입니다.

Description

CiAMA 클래스는 적응형 이동 평균 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiAMA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAMA

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
FastEmaPeriod	빠른 EMA의 평균 주기를 반환합니다
SlowEmaPeriod	느린 EMA의 평균 주기를 반환합니다
IndShift	수평 이동을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 요소를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

빠른 EMA의 평균 주기를 반환합니다.

```
int  FastEmaPeriod() const
```

Return Value

지표 생성 시 정의된 빠른 EMA의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

느린 EMA의 평균 주기를 반환합니다.

```
int  SlowEmaPeriod() const
```

Return Value

지표 생성 시 정의된 느린 EMA의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

IndShift

지표의 수평 이동을 반환합니다.

```
int  IndShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           string,              // 심볼
   ENUM_TIMEFRAMES  period,              // 주기
   int              ma_period,           // 평균화 주기
   int              fast_ema_period,     // 빠른 EMA 주기
   int              slow_ema_period,     // 느린 EMA 주기
   int              ind_shift,           // shift
   int              applied              // 가격 유형, 핸들
   )
```

Parameters

string
  [in] 심볼.

period
  [in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period
  [in] 평균화 주기.

fast_ema_period
  [in] 빠른 EMA 평균화 주기.

slow_ema_period
  [in] 느린 EMA 평균화 주기.

ind_shift
  [in] 수평 시프트.

applied
  [in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiAMA용 IND_AMA).

© 2000-2025, MetaQuotes Ltd.

CiVIDyA

CiVIDyA는 가변 지수 동적 평균 기술 지표를 사용하기 위한 클래스입니다.

Description

CiVIDyA 클래스는 가변 인덱스 동적 평균 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiVIDyA: public CIndicator
```

Title

```
#include <Indicators\Trend.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVIDyA

그룹별 클래스 메서드

Attributes	
CmoPeriod	운동량에 대한 기간을 반환합니다
EmaPeriod	평균 주기를 반환합니다
IndShift	수평 이동을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
메서드 생성	
Create	지표를 작성합니다
데이터 액세스 메서드	
Main	버퍼 요소를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

CmoPeriod

클래스 인스턴스에 바인딩된 차트를 닫습니다.

```
int  CmoPeriod() const
```

© 2000-2025, MetaQuotes Ltd.

EmaPeriod

차트 심볼을 반환합니다.

```
int  EmaPeriod() const
```

Return Value

클래스 인스턴스에 바인딩된 차트 심볼을 반환합니다. "" – 제한 없는 차트.

© 2000-2025, MetaQuotes Ltd.

IndShift

지표의 수평 이동을 반환합니다.

```
int  IndShift() const
```

Return Value

지표 작성 시 정의된 수평 이동 값을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,          // 심볼
   ENUM_TIMEFRAMES  period,          // 주기
   int              cmo_period,      // 모멘텀 주기
   int              ema_period,      // 평균화 주기
   int              ind_shift,       // 시프트
   int              applied          // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

cmo_period

[in] 모멘텀 기간.

ema_period

[in] 평균화 주기.

ind_shift

[in] 수평 시프트.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiVIDyA용 IND_VIDYA).

© 2000-2025, MetaQuotes Ltd.

Oscillator Classes

이 챕터 그룹에는 MQL5 표준 라이브러리의 해당 구성 요소에 대한 설명과 함께 오실레이터 클래스의 기술 세부 정보가 포함되어 있습니다.

클래스/그룹	Description
CiATR	평균 실제 범위
CiBearsPower	Bears Power
CiBullsPower	Bulls Power
CiCCI	물품 채널 색인
CiChaikin	Chaikin Oscillator
CiDeMarker	DeMarker
CiForce	Force Index
CiMACD	이동 평균 수렴-분산
CiMomentum	Momentum
CiOsMA	오실레이터 이동 평균 (MACD 히스토그램)
CiRSI	상대적 강세 지수
CiRVI	상대적 활력 지수
CiStochastic	확률적 오실레이터
CiWPR	윌리엄스 백분율 범위
CiTriX	삼중 지수 이동 평균 오실레이터

© 2000-2025, MetaQuotes Ltd.

CiATR

CiATR은 평균 실제 범위(Average True Range) 기술 지표를 사용하기 위한 클래스입니다.

Description

CiATR 클래스는 평균 실제 범위 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiATR: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiATR

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균화 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiATR용 IND_ATR).

© 2000-2025, MetaQuotes Ltd.

CiBearsPower

CiBearsPower는 베어스 파워 기술 지표를 사용하기 위한 클래스입니다.

Description

CiBearsPower 클래스는 베어스 파워 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiBearsPower: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBearsPower

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균화 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(IND_BEARS for CiBearsPower).

© 2000-2025, MetaQuotes Ltd.

CiBullsPower

CiBullsPower는 불스 파워 기술 지표를 사용하기 위한 클래스입니다.

Description

CiBullsPower 클래스는 불스 파워 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiBullsPower: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBullsPower

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 요소를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균화 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiBullsPower용 IND_BULLS).

© 2000-2025, MetaQuotes Ltd.

CiCCI

CiCCI는 상품 채널 지수 기술 지표를 사용하기 위한 클래스입니다.

Description

CiCCI 클래스는 물품 채널 인덱스 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiCCI: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiCCI

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(IND_CCI for CiCCI).

© 2000-2025, MetaQuotes Ltd.

CiChaikin

CiChaikin은 Chaikin 오실레이터 기술 지표를 사용하기 위한 클래스입니다.

Description

CiChaikin 클래스는 Chaikin 오실레이터 지표의 데이터 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiChaikin: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiChaikin

그룹별 클래스 메서드

Attributes	
FastMaPeriod	빠른 MA의 평균 주기를 반환합니다
SlowMaPeriod	느린 MA의 평균 주기를 반환합니다
MaMethod	평균화 방법을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastMaPeriod

빠른 EMA의 평균 주기를 반환합니다.

```
int  FastMaPeriod() const
```

Return Value

지표 생성 시 정의된 빠른 EMA의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SlowMaPeriod

느린 EMA의 평균 주기를 반환합니다.

```
int  SlowMaPeriod() const
```

Return Value

지표 생성 시 정의된 느린 EMA의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaMethod

평균화 메서드를 반환합니다

```
ENUM_MA_METHOD  MaMethod() const
```

Return Value

지표 작성 시 정의된 평균화 방법을 반환합니다(ENUM_MA_METHOD 열거값).

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 개체(볼륨 유형)를 반환합니다.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Return Value

적용할 개체(볼륨 유형), 지표 작성 시 정의됨 (ENUM_APPLIED_VOLUME 열거 값).

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string               symbol,              // 심볼
   ENUM_TIMEFRAMES      period,              // 주기
   int                  fast_ma_period,      // 빠른 EMA 주기
   int                  slow_ma_period,      // 느린 EMA 주기
   ENUM_MA_METHOD       ma_method,           // 평균화 메서드
   ENUM_APPLIED_VOLUME  applied              // 볼륨 유형
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

fast_ma_period

[in] 빠른 EMA용 주기.

slow_ma_period

[in] 느린 EMA용 주기.

ma_method

[in] 평균화 메서드 (ENUM_MA_METHOD 열거값).

applied

[in] 적용할 객체(볼륨 유형) (ENUM_APPLIED_VOLUME 열거 값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiChaikin용 IND_CHAIKIN).

© 2000-2025, MetaQuotes Ltd.

CiDeMarker

CiDeMarker는 DeMarker 기술 지표를 사용하기 위한 클래스입니다.

Description

CiDeMarker 클래스는 DeMarker 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiDeMarker: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiDeMarker

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균화 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiDeMarker용 IND_DEMARKER).

© 2000-2025, MetaQuotes Ltd.

CiForce

CiForce는 Force Index 기술 지표를 사용하기 위한 클래스입니다.

Description

CiForce 클래스는 Force Index 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiForce: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiForce

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
MaMethod	평균화 방법을 반환합니다
Applied	적용할 개체(볼륨 유형)를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

MaPeriod

平均化 周期を返します.

```
int  MaPeriod() const
```

Return Value

指標 生成 時 定義された 平均 周期を 返します.

© 2000-2025, MetaQuotes Ltd.

MaMethod

평균화 메서드를 반환합니다

```
ENUM_MA_METHOD  MaMethod() const
```

Return Value

지표 작성 시 정의된 평균화 방법을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 개체(볼륨 유형)를 반환합니다.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Return Value

적용할 개체(볼륨 유형), 지표 작성 시 정의됨 (ENUM_APPLIED_VOLUME 열거 값).

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string               symbol,        // 심볼
   ENUM_TIMEFRAMES      period,        // 주기
   int                  ma_period,     // 평균화 주기
   ENUM_MA_METHOD       ma_method,     // 평균화 메서드
   ENUM_APPLIED_VOLUME  applied        // 볼륨 유형
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

ma_method

[in] 평균화 메서드 (ENUM_MA_METHOD 열거값).

applied

[in] 적용할 객체(볼륨 유형) (ENUM_APPLIED_VOLUME 열거 값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiForce용 IND_FORCE).

© 2000-2025, MetaQuotes Ltd.

CiMACD

CiMACD는 이동평균 수렴-확산 기술지표를 사용하기 위한 클래스입니다.

Description

CiMACD 클래스는 이동 평균 수렴-발산 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiMACD: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMACD

그룹별 클래스 메서드

Attributes	
FastEmaPeriod	빠른 EMA의 평균 주기를 반환합니다
SlowEmaPeriod	느린 EMA의 평균 주기를 반환합니다
SignalPeriod	시그널 라인의 평균 주기를 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	메인 선의 버퍼 데이터를 반환합니다
Signal	시그널 라인의 버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

빠른 EMA의 평균 주기를 반환합니다.

```
int  FastEmaPeriod() const
```

Return Value

지표 생성 시 정의된 빠른 EMA의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

느린 EMA의 평균 주기를 반환합니다.

```
int  SlowEmaPeriod() const
```

Return Value

지표 생성 시 정의된 느린 EMA의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SignalPeriod

시그널 라인의 평균 주기를 반환합니다.

```
int  SignalPeriod() const
```

Return Value

지표 생성 시 정의된 시그널 라인의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,              // 심볼
   ENUM_TIMEFRAMES  period,              // 주기
   int              fast_ema_period,     // 빠른 EMA 주기
   int              slow_ema_period,     // 느린 EMA 주기
   int              signal_period,       // 시그널 주기
   int              applied              // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

fast_ema_period

[in] 빠른 EMA 평균화 주기.

slow_ema_period

[in] 느린 EMA 평균화 주기.

signal_period

[in] 시그널 라인 평균화 주기.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스로 메인 선 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스의 주선 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Signal

시그널 라인의 버퍼 요소를 지정한 인덱스만큼 반환합니다.

```
double  Signal(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 신호 라인의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiMACD용 IND_MACD).

© 2000-2025, MetaQuotes Ltd.

CiMomentum

CiMonentum은 Motimum 기술 지표를 사용하기 위한 클래스입니다.

Description

CiMonentum 클래스는 Motimum 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiMomentum: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMomentum

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Applied	적용할 개체(볼륨 유형)를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 요소를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiMomentum용 IND_MOMENTUM).

© 2000-2025, MetaQuotes Ltd.

CiOsMA

CiOsMA는 MACD 히스토그램(오실레이터의 이동 평균) 기술 지표를 사용하기 위한 클래스입니다.

Description

CiOsMA 클래스는 MACD 히스토그램(오실레이터의 이동 평균) 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiOsMA: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOsMA

그룹별 클래스 메서드

Attributes	
FastEmaPeriod	빠른 EMA의 평균 주기를 반환합니다
SlowEmaPeriod	느린 EMA의 평균 주기를 반환합니다
SignalPeriod	시그널 라인의 평균 주기를 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 요소를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

FastEmaPeriod

빠른 EMA의 평균 주기를 반환합니다.

```
int  FastEmaPeriod() const
```

Return Value

지표 생성 시 정의된 빠른 EMA의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SlowEmaPeriod

느린 EMA의 평균 주기를 반환합니다.

```
int  SlowEmaPeriod() const
```

Return Value

지표 생성 시 정의된 느린 EMA의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

SignalPeriod

시그널 라인의 평균 주기를 반환합니다.

```
int  SignalPeriod() const
```

Return Value

지표 생성 시 정의된 시그널 라인의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,              // 심볼
   ENUM_TIMEFRAMES  period,              // 주기
   int              fast_ema_period,     // 빠른 EMA 주기
   int              slow_ema_period,     // 느린 EMA 주기
   int              signal_period,       // 시그널 라인 주기
   int              applied              // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

fast_ema_period

[in] 빠른 EMA 평균화 주기.

slow_ema_period

[in] 느린 EMA 평균화 주기.

signal_period

[in] 시그널 라인 평균화 주기.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiOsMA용 IND_OSMA).

© 2000-2025, MetaQuotes Ltd.

CiRSI

CiRSIは 상대강도지수 기술지표를 사용하기 위한 클래스입니다.

Description

CiRSI 클래스는 상대 강도 지수 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiRSI: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRSI

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 요소를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

applied

[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiRSI용 IND_RSI).

© 2000-2025, MetaQuotes Ltd.

CiRVI

CiRVI는 상대 활력 지수 기술 지표를 사용하기 위한 클래스입니다.

Description

CiRVI 클래스는 상대적 활력 지수 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiRVI: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRVI

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
메서드 생성	
Create	지표를 작성합니다
데이터 액세스 메서드	
Main	메인 선의 버퍼 데이터를 반환합니다
Signal	시그널 라인의 버퍼 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period      // 평균화 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스로 메인 선 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 지수.

Return Value

지정된 인덱스의 주선 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Signal

시그널 라인의 버퍼 요소를 지정한 인덱스만큼 반환합니다.

```
double  Signal(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 신호 라인의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiRVI용 IND_RVI).

© 2000-2025, MetaQuotes Ltd.

CiStochastic

CiStochastic은 확률적 오실레이터 기술 지표를 사용하기 위한 클래스입니다.

Description

CiStochastic 클래스는 확률적 오실레이터 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiStochastic: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiStochastic

그룹별 클래스 메서드

Attributes	
Kperiod	%K 라인의 평균 주기를 반환합니다
Dperiod	%D 라인의 평균 기간을 반환합니다
Slowing	느린 기간을 반환합니다
MaMethod	평균화 방법을 반환합니다
PriceField	적용할 가격 유형(저가/고가 또는 종가/시가)
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	메인 선의 버퍼 데이터를 반환합니다
Signal	시그널 라인의 버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Kperiod

%K 라인에 대한 평균 주기를 반환합니다.

```
int  Kperiod() const
```

Return Value

지표 작성 시 정의된 %K 라인에 대한 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Dperiod

%D 라인에 대한 평균 주기를 반환합니다.

```
int  Dperiod() const
```

Return Value

지표 작성 시 정의된 %D 라인에 대한 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Slowing

느려지는 기간을 반환합니다.

```
int  Slowing() const
```

Return Value

지표 작성 시 정의된 감속 기간을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MaMethod

평균화 메서드를 반환합니다

```
ENUM_MA_METHOD  MaMethod() const
```

Return Value

지표 작성 시 정의된 평균화 방법을 반환합니다(ENUM_MA_METHOD 열거값).

© 2000-2025, MetaQuotes Ltd.

PriceField

적용할 객체(저가/고가 또는 종가/시가)를 반환합니다.

```
ENUM_STO_PRICE  PriceField() const
```

Return Value

적용할 객체(저가/고가 또는 종가/시가) 로, 지표 작성시 정의됨(ENUM_STO_PRICE 열거값).

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,          // 심볼
   ENUM_TIMEFRAMES  period,          // 주기
   int              Kperiod,         // %K 주기
   int              Dperiod,         // %D 주기
   int              slowing,         // 느린 주기
   ENUM_MA_METHOD   ma_method,       // 평균화 메서드
   ENUM_STO_PRICE   price_field      // 어플리케이션
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

Kperiod

[in] %K 지표의 평균화 기간.

Dperiod

[in] %D 지표의 평균화 기간.

slowing

[in] Slowing period.

ma_method

[in] 평균화 메서드 (ENUM_MA_METHOD 열거값).

price_field

[in] 적용할 객체(저가/고가 또는 종가/시가) (ENUM_STO_PRICE 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스로 메인 선 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스의 주선 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Signal

시그널 라인의 버퍼 요소를 지정한 인덱스만큼 반환합니다.

```
double  Signal(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 신호 라인의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiStochastic용 IND_STOCHASTIC).

© 2000-2025, MetaQuotes Ltd.

CiTriX

CiTriX는 삼중 지수 이동 평균 오실레이터 기술 지표를 사용하기 위한 클래스입니다.

Description

CiTriX 클래스는 삼중 지수 이동 평균 오실레이터 지표의 생성, 설정 및 데이터 액세스를 제공합니다.

Declaration

```
class CiTriX: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTriX

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,        // 심볼
   ENUM_TIMEFRAMES  period,        // 주기
   int              ma_period,     // 평균화 주기
   int              applied        // 가격 유형, 핸들
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

applied

[in] 적용할 핸들의 가격 유형.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiTriX용 IND_TRIX).

© 2000-2025, MetaQuotes Ltd.

CiWPR

CiWPR은 윌리엄스 백분율 범위 기술 지표를 사용하기 위한 클래스입니다.

Description

CiWPR 클래스는 Williams의 Percent Range 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiWPR: public CIndicator
```

Title

```
#include <Indicators\Oscilators.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiWPR

그룹별 클래스 메서드

Attributes	
CalcPeriod	계산 주기를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

CalcPeriod

계산할 기간을 반환합니다.

```
int  CalcPeriod() const
```

Return Value

지표 작성 시 정의된 계산 기간을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,          // 심볼
   ENUM_TIMEFRAMES  period,          // 주기
   int              calc_period      // 계산 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

calc_period

[in] 계산 주기.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiWPR용 IND_WPR).

© 2000-2025, MetaQuotes Ltd.

Volume Indicators

이 챕터 그룹에는 볼륨 표시기 클래스의 기술적 세부 정보와 MQL5 표준 라이브러리의 모든 해당 핵심 구성요소에 대한 설명이 포함되어 있습니다.

클래스/그룹	Description
CiAD	수렴/발산
CiMFI	자금 흐름 지수
CiOBV	On Balance Volume
CiVolumes	Volumes

© 2000-2025, MetaQuotes Ltd.

CiAD

CiAD는 수렴/발산 기술 지표를 사용하기 위한 클래스입니다.

Description

CiAD 클래스는 수렴/발산 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiAD: public CIndicator
```

Title

```
#include <Indicators\Volumes.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAD

그룹별 클래스 메서드

Attributes	
Applied	계산 주기를 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 볼륨 유형을 반환합니다.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Return Value

적용할 볼륨 유형, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string                symbol,      // 심볼
   ENUM_TIMEFRAMES       period,      // 주기
   ENUM_APPLIED_VOLUME   applied      // 볼륨 유형
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

applied

[in] 적용할 볼륨 유형(ENUM_APPLIED_VOLUME 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiAD용 IND_AD).

© 2000-2025, MetaQuotes Ltd.

CiMFI

CiMFI는 자금 흐름 지수 기술 지표를 사용하기 위한 클래스입니다.

Description

CiMFI 클래스는 자금 흐름 지수 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiMFI: public CIndicator
```

Title

```
#include <Indicators\Volumes.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMFI

그룹별 클래스 메서드

Attributes	
MaPeriod	평균 주기를 반환합니다
Applied	적용할 볼륨 유형을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

MaPeriod

평균화 주기를 반환합니다.

```
int  MaPeriod() const
```

Return Value

지표 생성 시 정의된 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 볼륨 유형을 반환합니다.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Return Value

적용할 볼륨 유형, 지표 생성 시 정의됩니다(ENUM_APPLIED_VOLUME 열거 값).

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string               symbol,        // 심볼
   ENUM_TIMEFRAMES      period,        // 주기
   int                  ma_period,     // 평균화 주기
   ENUM_APPLIED_VOLUME  applied        // 볼륨 유형
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

ma_period

[in] 평균화 주기.

applied

[in] 적용할 볼륨 유형(ENUM_APPLIED_VOLUME 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000-2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiMFI용 IND_MFI).

© 2000-2025, MetaQuotes Ltd.

CiOBV

CiOBV는 On Balance Volume 기술 지표를 사용하기 위한 클래스입니다.

Description

CiOBV 클래스는 볼륨 균형 조정 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiOBV: public CIndicator
```

Title

```
#include <Indicators\Volumes.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOBV

그룹별 클래스 메서드

Attributes	
Applied	적용할 볼륨 유형을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Applied

적용할 볼륨 유형을 반환합니다.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Return Value

적용할 볼륨 유형, 지표 생성 시 정의됩니다(ENUM_APPLIED_VOLUME 열거 값).

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string                symbol,      // 심볼
   ENUM_TIMEFRAMES       period,      // 주기
   ENUM_APPLIED_VOLUME   applied      // 볼륨 유형
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

applied

[in] 적용할 볼륨 유형(ENUM_APPLIED_VOLUME 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiOBV용 IND_OBV).

© 2000-2025, MetaQuotes Ltd.

CiVolumes

CiVolumes는 Volumes 기술 지표를 사용하기 위한 클래스입니다.

Description

CiVolumes 클래스는 Volumes 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiVolumes: public CIndicator
```

Title

```
#include <Indicators\Volumes.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVolumes

그룹별 클래스 메서드

Attributes	
Applied	적용할 볼륨 유형을 반환합니다
메서드 생성	
Create	지표를 작성합니다
데이터 액세스 메서드	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Applied

적용할 볼륨 유형을 반환합니다.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Return Value

적용할 볼륨 유형, 지표 생성 시 정의됩니다(ENUM_APPLIED_VOLUME 열거 값).

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string                symbol,      // 심볼
   ENUM_TIMEFRAMES       period,      // 주기
   ENUM_APPLIED_VOLUME   applied      // 볼륨 유형
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

applied

[in] 적용할 볼륨 유형(ENUM_APPLIED_VOLUME 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiVolumes용 IND_VOLUMES).

© 2000-2025, MetaQuotes Ltd.

Bill Williams Indicators

이 챕터 그룹에는 Bill Williams 지표 클래스의 기술적 세부 사항과 MQL5 표준 라이브러리의 모든 해당 구성 요소에 대한 설명이 포함되어 있다.

클래스/그룹	Description
CiAC	가속기 오실레이터
CiAlligator	Alligator
CiAO	Awesome 오실레이터
CiFractals	Fractals
CiGator	게이터 오실레이터
CiBWMFI	시장 촉진 지수

© 2000-2025, MetaQuotes Ltd.

CiAC

CiAC는 가속기 오실레이터 기술 표시를 사용하기 위한 클래스입니다.

Description

CiAC 클래스는 가속기 오실레이터 지표의 데이터 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiAC: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAC

그룹별 클래스 메서드

Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

식별자 타입(CiAC용 IND_AC).

© 2000-2025, MetaQuotes Ltd.

CiAlligator

CiAlligator는 앨리게이터 기술 지표를 사용하기 위한 클래스입니다.

Description

CiAlligator 클래스는 앨리게이터 지표 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiAlligator: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAlligator

그룹별 클래스 메서드

Attributes	
JawPeriod	턱 선의 평균 기간을 반환합니다
JawShift	턱 선의 수평 이동을 반환합니다
TeethPeriod	이빨 선에 대한 평균 기간을 반환합니다
TeethShift	이빨 선의 수평 이동을 반환합니다
LipsPeriod	입술 선에 대한 평균 기간을 반환합니다
LipsShift	입술 선의 수평 이동을 반환합니다
MaMethod	평균화 방법을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Jaw	턱 선 버퍼의 버퍼 데이터를 반환합니다
Teeth	이빨 선 버퍼의 버퍼 데이터를 반환합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
Lips	입술 선 버퍼의 버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

JawPeriod

턱 라인에 대한 평균 주기를 반환합니다.

```
int  JawPeriod() const
```

Return Value

지표 작성 시 정의된 턱 라인에 대한 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

JawShift

턱 선의 수평 이동을 반환합니다.

```
int  JawShift() const
```

Return Value

턱 선의 수평 이동은 지표 생성시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

TeethPeriod

치아 선에 대한 평균 기간을 반환합니다.

```
int  TeethPeriod() const
```

Return Value

그 치아 선에는 지표 창작품에 정의된 그 평균화 기간을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TeethShift

치아 선의 수평 이동을 반환합니다.

```
int  TeethShift() const
```

Return Value

지표 생성 시 정의된 치아 선의 수평 이동.

© 2000-2025, MetaQuotes Ltd.

LipsPeriod

입술 선의 평균 기간을 반환합니다.

```
int  LipsPeriod() const
```

Return Value

지표 작성 시 정의된 입술 선의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

LipsShift

입술 선의 수평 이동을 반환합니다.

```
int  LipsShift() const
```

Return Value

입술 선의 수평 이동은 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

MaMethod

평균화 메서드를 반환합니다

```
ENUM_MA_METHOD  MaMethod() const
```

Return Value

지표 작성 시 정의된 평균화 방법을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,           // 심볼
   ENUM_TIMEFRAMES  period,           // 주기
   int              jaw_period,       // 턱 주기
   int              jaw_shift,        // 턱 시프트
   int              teeth_period,     // 이빨 주기
   int              teeth_shift,      // 이빨 시프트
   int              lips_period,      // 입술 주기
   int              lips_shift,       // 입술 시프트
   ENUM_MA_METHOD   ma_method,        // 평균 메서드
   int              applied           // 가격 유형, 핸들러
   )
```

Parameters

symbol
[in] 심볼.

period
[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

jaw_period
[in] 턱 평균 주기.

jaw_shift
[in] 턱 수평 시프트.

teeth_period
[in] 이빨 평균 주기.

teeth_shift
[in] 이빨 수평 시프트.

lips_period
[in] 입술 평균 주기.

lips_shift
[in] 입술 수평 시프트.

ma_method
[in] 이동 평균 메서드(ENUM_MA_METHOD 열거 값).

applied
[in] 가격 유형, 적용할 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Jaw

턱 선의 버퍼 요소를 지정한 인덱스별로 반환합니다.

```
double  Jaw(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 턱 선 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 턱 선의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Teeth

이빨 선의 버퍼 요소를 지정한 인덱스만큼 반환합니다.

```
double  Teeth(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 톱니 라인 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 Teas 선의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Lips

입술 선의 버퍼 요소를 지정한 인덱스별로 반환합니다.

```
double  Lips(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 입술 선 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 입술 선의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE입니다.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiAlligator용 IND_ALLIGATOR).

© 2000-2025, MetaQuotes Ltd.

CiAO

CiAO는 Awesome 오실레이터 기술 지표를 사용하기 위한 클래스입니다.

Description

CiAO 클래스는 Awesome 오실레이터 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiAO: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAO

그룹별 클래스 메서드

Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 요소를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스의 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiAO IND_AO).

© 2000-2025, MetaQuotes Ltd.

CiFractals

CiFractals는 프랙탈 기술 지표를 사용하기 위한 클래스입니다.

Description

CiFractals 클래스는 프랙탈 지표의 데이터에 대한 생성, 설정 및 액세스를 제공합니다.

Declaration

```
class CiFractals: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFractals

그룹별 클래스 메서드

Create	
Create	지표를 작성합니다
데이터 액세스	
Upper	위쪽 버퍼의 데이터를 반환합니다
Lower	하위 버퍼의 데이터를 반환합니다
Input/output	
virtual Type	가상 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,     // 심볼
   ENUM_TIMEFRAMES  period      // 주기
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Upper

지정한 인덱스별로 위쪽 버퍼의 요소를 반환합니다.

```
double  Upper(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 상위 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 상위 버퍼의 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Lower

지정된 인덱스별로 하위 버퍼의 요소를 반환합니다.

```
double  Lower(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 하위 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 하부 버퍼의 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiFractals용 IND_FRACTALS).

© 2000-2025, MetaQuotes Ltd.

CiGator

CiGator는 Gator 오실레이터 기술 지표를 사용하기 위한 클래스입니다.

Description

CiGator 클래스는 Gator 오실레이터 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiGator: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

상속 계층

CObject
CArray
CArrayObj
CSeries
CIndicator
CiGator

그룹별 클래스 메서드

Attributes	
JawPeriod	턱 선의 평균 기간을 반환합니다
JawShift	턱 선의 수평 이동을 반환합니다
TeethPeriod	이빨 선에 대한 평균 기간을 반환합니다
TeethShift	이빨 선의 수평 이동을 반환합니다
LipsPeriod	입술 선에 대한 평균 기간을 반환합니다
LipsShift	입술 선의 수평 이동을 반환합니다
MaMethod	평균화 방법을 반환합니다
Applied	적용할 가격 유형 또는 핸들을 반환합니다
메서드 생성	
Create	지표를 작성합니다
데이터 액세스 메서드	
Upper	위쪽 버퍼의 데이터를 반환합니다
Lower	하위 버퍼의 데이터를 반환합니다

© 2000-2025, MetaQuotes Ltd.

Attributes	
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000-2025, MetaQuotes Ltd.

JawPeriod

죠스 선의 평균 기간을 반환합니다.

```
int  JawPeriod() const
```

Return Value

지표 작성 시 정의된 죠스 선의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

JawShift

턱 선의 수평 이동을 반환합니다.

```
int  JawShift() const
```

Return Value

턱 선의 수평 이동은 지표 생성시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

TeethPeriod

치아 선에 대한 평균 기간을 반환합니다.

```
int  TeethPeriod() const
```

Return Value

그 치아 선에는 지표 창작품에 정의된 그 평균화 기간을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TeethShift

치아 선의 수평 이동을 반환합니다.

```
int  TeethShift() const
```

Return Value

지표 생성 시 정의된 치아 선의 수평 이동.

© 2000-2025, MetaQuotes Ltd.

LipsPeriod

입술 선의 평균 기간을 반환합니다.

```
int  LipsPeriod() const
```

Return Value

지표 작성 시 정의된 입술 선의 평균 주기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

LipsShift

입술 선의 수평 이동을 반환합니다.

```
int  LipsShift() const
```

Return Value

입술 선의 수평 이동은 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

MaMethod

평균화 메서드를 반환합니다

```
ENUM_MA_METHOD  MaMethod() const
```

Return Value

지표 작성 시 정의된 평균화 방법을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Applied

적용할 가격 유형 또는 핸들을 반환합니다.

```
int  Applied() const
```

Return Value

적용할 가격 유형 또는 핸들, 지표 생성 시 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string           symbol,           // 심볼
   ENUM_TIMEFRAMES  period,           // 주기
   int              jaw_period,       // 턱 주기
   int              jaw_shift,        // 턱 시프트
   int              teeth_period,     // 이빨 주기
   int              teeth_shift,      // 이빨 시프트
   int              lips_period,      // 입술 주기
   int              lips_shift,       // 입술 시프트
   ENUM_MA_METHOD   ma_method,        // 평균 메서드
   int              applied           // 가격 유형, 핸들러
   )
```

Parameters

symbol
[in] 심볼.

period
[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

jaw_period
[in] 턱 평균 주기.

jaw_shift
[in] 턱 수평 시프트.

teeth_period
[in] 이빨 평균 주기.

teeth_shift
[in] 이빨 수평 시프트.

lips_period
[in] 입술 평균 주기.

lips_shift
[in] 입술 수평 시프트.

ma_method
[in] 평균화 메서드 (ENUM_MA_METHOD 열거값).

applied
[in] 적용할 가격 유형 또는 핸들.

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Upper

지정한 인덱스별로 위쪽 버퍼의 요소를 반환합니다.

```
double  Upper(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 상위 버퍼 요소 색인.

Return Value

지정된 인덱스의 위쪽 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Lower

지정된 인덱스별로 하위 버퍼의 요소를 반환합니다.

```
double  Upper(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 하위 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 하부 버퍼의 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiGator용 IND_GATOR).

© 2000-2025, MetaQuotes Ltd.

CiBWMFI

CiBWMFI는 빌 윌리엄스 기술 지표의 시장 촉진 지수를 사용하기 위한 클래스입니다.

Description

CiBWMFI 클래스는 Bill Williams 지표에 의한 시장 촉진 지수의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiBWMFI: public CIndicator
```

Title

```
#include <Indicators\BillWilliams.mqh>
```

상속 계층

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBWMFI

그룹별 클래스 메서드

Attributes	
Applied	적용할 볼륨 유형을 반환합니다
Create	
Create	지표를 작성합니다
데이터 액세스	
Main	버퍼 데이터를 반환합니다
Input/output	
virtual Type	가삭 식별 메서드

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

클래스 CArray에서 상속된 메서드

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

CArrayObj 클래스에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Compare

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

클래스 CSeries에서 상속된 메서드

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

클래스 CIndicator에서 상속된 메서드

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

© 2000–2025, MetaQuotes Ltd.

Applied

적용할 볼륨 유형을 반환합니다.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Return Value

적용할 볼륨 유형, 지표 생성 시 정의됩니다(ENUM_APPLIED_VOLUME 열거 값).

© 2000-2025, MetaQuotes Ltd.

Create

지정된 매개변수로 지표를 작성합니다. Refresh() 및 GetData()를 사용하여 지표 값을 업데이트하고 가져옵니다.

```
bool  Create(
   string                symbol,      // 심볼
   ENUM_TIMEFRAMES       period,      // 주기
   ENUM_APPLIED_VOLUME   applied      // 볼륨 유형
   )
```

Parameters

symbol

[in] 심볼.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

applied

[in] 적용할 볼륨 유형(ENUM_APPLIED_VOLUME 열거값).

Return Value

true – 성공, false – 지표를 만들 수 없음.

© 2000–2025, MetaQuotes Ltd.

Main

지정한 인덱스별로 버퍼 요소를 반환합니다.

```
double  Main(
   int  index      // 인덱스
   )
```

Parameters

index

[in] 버퍼 요소 색인.

Return Value

지정된 인덱스에 의한 버퍼 요소 또는 올바른 데이터가 없는 경우 EMPTY_VALUE.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiBWMFI용 IND_BWMFI).

© 2000-2025, MetaQuotes Ltd.

CiCustom

CiCustom은 사용자 정의 기술 지표를 사용하기 위한 클래스입니다.

Description

CiCustom 클래스는 커스텀 지표의 생성, 설정 및 데이터에 대한 액세스를 제공합니다.

Declaration

```
class CiCustom: public CIndicator
```

Title

```
#include <Indicators\Custom.mqh>
```

그룹별 클래스 메서드

Attributes	
NumBuffers	버퍼의 수를 설정합니다
NumParams	지표를 작성할 때 사용되는 매개변수 수를 가져옵니다
ParamType	지정된 매개변수의 유형을 가져옵니다
ParamLong	정수 유형의 지정된 매개변수 값을 가져옵니다
ParamDouble	double 유형의 지정된 매개변수 값을 가져옵니다
ParamString	문자열 유형의 지정된 매개변수 값을 가져옵니다
Input/output	
virtual Type	가삭 식별 메서드

© 2000-2025, MetaQuotes Ltd.

NumBuffers

버퍼 수를 설정합니다.

```
bool  NumBuffers(
```

Return Value

true – 성공, false – 필요한 버퍼 수를 설정할 수 없음.

© 2000-2025, MetaQuotes Ltd.

NumParams

지표를 작성할 때 사용되는 매개변수 수를 가져옵니다.

```
int  NumParams() const
```

Return Value

지표를 만드는 데 사용되는 매개변수 수.

© 2000-2025, MetaQuotes Ltd.

ParamType

매개변수의 유형을 가져옵니다.

```
ENUM_DATATYPE  ParamType(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 매개변수 색인.

Return Value

지표 작성에 사용되는 지정된 매개 변수의 데이터 형식을 반환합니다.

Note

매개 변수 인덱스가 올바르지 않으면 WRONG_VALUE를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

ParamLong

long 유형의 지정된 매개변수 값을 가져옵니다.

```
long  ParamLong(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 매개변수 색인.

Return Value

지표를 만드는 데 사용되는 긴 형식의 지정된 매개 변수 값입니다.

Note

매개 변수 인덱스가 잘못되었거나 매개 변수 유형이 길지 않으면 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

ParamDouble

이중 유형의 지정된 매개변수 값을 가져옵니다.

```
double  ParamDouble(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 매개변수 색인.

Return Value

지표 작성에 사용되는 double 유형의 지정된 매개 변수 값.

Note

매개 변수 인덱스가 잘못되었거나 매개 변수 형식이 double 형식이 아니면 EMPTY_VALUE를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

ParamString

문자열 유형의 지정된 매개변수 값을 가져옵니다.

```
string  ParamString(
   int  index      // 인덱스
   ) const
```

Parameters

index

[in] 매개변수 색인.

Return Value

지표를 만드는 데 사용되는 지정된 문자열 매개 변수의 값.

Note

숫자가 잘못되었거나 매개 변수가 문자열 유형이 아니면 빈 문자열을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Type

가상 식별 메서드.

```
virtual int  Type() const
```

Return Value

지표 유형(CiCustom용 IND_CUSTOM).

© 2000-2025, MetaQuotes Ltd.

트레이드 클래스

이 섹션에는 MQL5 스탠다드 라이브러리의 관련 구성 요소에 대한 설명과 트레이드 클래스 작업에 대한 기술 세부 정보가 포함되어 있습니다.

트레이드 클래스를 사용하면 사용자 지정 프로그램(엑스퍼트 어드바이저)을 만들 때 시간이 절약됩니다.

MQL5 스탠다드 라이브러리(트레이드 클래스의 경우)는 Include₩Trade 폴더의 터미널 작업 디렉토리에 배치됩니다.

클래스/그룹	설명
CAccountInfo	트레이드 계정 속성 관련 작업을 위한 클래스
CSymbolInfo	상품 속성 관련 작업 클래스
COrderInfo	지정가 주문 속성 관련 작업을 위한 클래스
CHistoryOrderInfo	내역 순서 속성 관련 작업을 위한 클래스
CPositionInfo	오픈 포지션 속성으로 작업하기 위한 클래스
CDealInfo	내역 딜 속성 작업을 위한 클래스
CTrade	트레이드 작업 실행을 위한 클래스
CTerminalInfo	터미널 환경의 속성을 가져오는 클래스

© 2000-2025, MetaQuotes Ltd.

CAccountInfo

CAccountInfo는 현재 열려 있는 거래 계정 속성에 쉽게 액세스할 수 있는 클래스입니다.

설명

CAccountInfo 클래스는 현재 오픈되어 있는 거래 계정 속성에 쉽게 액세스할 수 있도록 합니다.

선언

```
class CAccountInfo : public CObject
```

제목

```
#include <Trade\AccountInfo.mqh>
```

상속 계층

CObject

CAccountInfo

그룹별 클래스 메서드

정수 유형 속성에 대한 액세스	
Login	계정 번호 가져오기
TradeMode	거래 모드 가져오기
TradeModeDescription	거래 모드를 문자열로 가져오기
Leverage	주어진 레버리지의 금액 가져오기
StopoutMode	강제 청산 설정 모드 가져오기
StopoutModeDescription	문자열로 강제 청산 설정 모드 가져오기
TradeAllowed	거래 허용 플래그 가져오기
TradeExpert	자동 거래 허용 플래그 가져오기
LimitOrders	허용되는 최대 지정가 주문 수 가져오기
MarginMode	마진 계산 모드 가져오기
MarginModeDescription	문자열로 마진 계산 모드 가져오기
double 유형 속성에 액세스	
Balance	계정 잔액 가져오기
Credit	지정된 크레딧 금액을 가져오기
Profit	계정의 경상 이익 금액 가져오기
Equity	계정의 현재 지분 금액 가져오기

© 2000-2025, MetaQuotes Ltd.

정수 유형 속성에 대한 액세스	
Margin	예약된 마진 양 가져오기
FreeMargin	여유 마진 금액 가져오기
MarginLevel	마진 수준 가져오기
MarginCall	예금 마진 수준 가져오기
MarginStopOut	강제 청산 마진 수준 가져오기
텍스트 설정에 액세스	
Name	클라이언트 이름 가져오기
Server	거래 서버 이름 가져오기
Currency	예금 통화 이름 가져오기
Company	계정을 제공하는 회사 이름 가져오기
MQL5 API 기능에 액세스	
InfoInteger	지정된 정수 유형 속성 값 가져오기
InfoDouble	지정된 double 유형 속성 값 가져오기
InfoString	지정된 string 유형 속성 값 가져오기
추가 메서드	
OrderProfitCheck	전달된 매개변수를 기준으로 평가된 수익 가져오기
MarginCheck	거래 작업을 실행하는 데 필요한 마진 양을 가져오기
FreeMarginCheck	거래 작업 실행 후 남은 여유 마진 양 가져오기
MaxLotCheck	가능한 최대 거래 작업 볼륨 가져오기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Login

계좌 번호 가져오기.

```
long  Login() const
```

값 반환

계좌 번호.

© 2000-2025, MetaQuotes Ltd.

TradeMode

거래 모드 가져오기.

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

값 반환

ENUM_ACCOUNT_TRADE_MODE 열거의 거래 모드.

© 2000-2025, MetaQuotes Ltd.

TradeModeDescription

거래 모드를 문자열로 가져오기.

```
string  TradeModeDescription() const
```

값 반환

문자열의 거래 모드.

© 2000-2025, MetaQuotes Ltd.

Leverage

주어진 레버리지 양 가져오기.

```
long  Leverage() const
```

값 반환

주어진 레버리지 양.

© 2000-2025, MetaQuotes Ltd.

StopoutMode

강제 청산 수준 사양의 모드 가져오기.

```
ENUM_ACCOUNT_STOPOUT_MODE  StopoutMode() const
```

값 반환

ENUM_ACCOUNT_STOPOUT_MODE 열거의 강제 청산 설정 모드.

© 2000-2025, MetaQuotes Ltd.

StopoutModeDescription

강제 청산 수준 사야의 모드를 문자열로 가져오기.

```
string  StopoutModeDescription() const
```

값 반환

강제 청산 모드를 문자열로 설정.

© 2000-2025, MetaQuotes Ltd.

MarginMode

마진 계산 모드 가져오기.

```
ENUM_ACCOUNT_MARGIN_MODE  MarginMode() const
```

값 반환

ENUM_ACCOUNT_MARGIN_MODE 열거에서의 마진 계산.

© 2000-2025, MetaQuotes Ltd.

MarginModeDescription

마진 계산 모드를 문자열로 가져오기.

```
string  MarginModeDescription() const
```

값 반환

문자열의 마진 계산 모드.

© 2000-2025, MetaQuotes Ltd.

TradeAllowed

거래 허용량의 플래그 가져오기.

```
bool  TradeAllowed() const
```

값 반환

거래 허용량의 플래그.

© 2000-2025, MetaQuotes Ltd.

TradeExpert

자동 거래 허용량의 플래그 가져오기.

```
bool  TradeExpert() const
```

값 반환

자동 거래 허용량 플래그.

© 2000-2025, MetaQuotes Ltd.

LimitOrders

허용되는 최대 지정가 주문 수 가져오기

```
int  LimitOrders() const
```

값 반환

허용되는 최대 지정가 주문 수.

참고

0 - 제한 없음.

© 2000-2025, MetaQuotes Ltd.

Balance

계좌 잔고 가져오기.

```
double  Balance() const
```

값 반환

계좌 잔고(예금 통화).

© 2000-2025, MetaQuotes Ltd.

Credit

지정된 크레딧 금액 가져오기.

```
double  Credit() const
```

값 반환

주어진 크레딧 금액 (예금 통화).

© 2000-2025, MetaQuotes Ltd.

Profit

계정의 경상 이익 금액 가져오기.

```
double  Profit() const
```

값 반환

계좌상 경상 이익 금액(예금 통화).

© 2000-2025, MetaQuotes Ltd.

Equity

계좌의 경상자본액 가져오기.

```
double  Equity() const
```

값 반환

계좌의 경상자본액(예금 통화).

© 2000-2025, MetaQuotes Ltd.

Margin

예약된 마진 금액 가져오기.

```
double  Margin() const
```

값 반환

예약된 마진 금액(예금 통화).

© 2000-2025, MetaQuotes Ltd.

FreeMargin

여유 마진 금액 가져오기.

```
double  FreeMargin() const
```

값 반환

여유 마진 금액 (예금 통화).

© 2000-2025, MetaQuotes Ltd.

MarginLevel

마진 수준 가져오기.

```
double  MarginLevel() const
```

값 반환

마진 수준.

© 2000-2025, MetaQuotes Ltd.

MarginCall

예금의 마진 수준 가져오기.

```
double  MarginCall() const
```

값 반환

예금의 마진 수준.

© 2000-2025, MetaQuotes Ltd.

MarginStopOut

강제 청산 마진 수준 가져오기.

```
double  MarginStopOut() const
```

값 반환

강제 청산 마진 수준.

© 2000-2025, MetaQuotes Ltd.

Name

클라이언트 이름 가져오기.

```
string  Name() const
```

값 반환

클라이언트 이름.

© 2000-2025, MetaQuotes Ltd.

Server

거래 서버 이름 가져오기.

```
string  Server() const
```

값 반환

거래 서버 이름.

© 2000-2025, MetaQuotes Ltd.

Currency

예금 통화 이름 가져오기.

```
string  Currency() const
```

값 반환

예금 통화 이름.

© 2000-2025, MetaQuotes Ltd.

Company

계정을 제공하는 회사 이름 가져오기.

```
string  Company() const
```

값 반환

계정을 제공하는 회사 이름.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

지정한 정수 유형 속성 값 가져오기.

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // 속성 ID
   ) const
```

매개변수

prop_id

[in] 속성 식별자. 값은 ENUM_ACCOUNT_INFO_INTEGER 열거 값 중 하나일 수 있습니다.

값 반환

long 유형의 값.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

지정된 double 유형 속성 값을 가져오기.

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // 속성 ID
   ) const
```

매개변수

prop_id

[in] 속성 식별자. 값은 ENUM_ACCOUNT_INFO_DOUBLE 열거 값 중 하나일 수 있습니다.

값 반환

double 유형 값.

© 2000-2025, MetaQuotes Ltd.

InfoString

지정한 문자열 유형 속성의 값을 가져오기.

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // 속성 ID
   ) const
```

매개변수

prop_id

[in] 속성 식별자. 값은 ENUM_ACCOUNT_INFO_STRING 열거 값 중 하나일 수 있습니다.

값 반환

string 유형의 값.

© 2000-2025, MetaQuotes Ltd.

OrderProfitCheck

함수는 전달된 매개변수를 기준으로 경상 계정의 이익을 계산합니다. 이 함수는 거래 작업 결과의 사전 평가에 사용됩니다. 이 값은 계정 통화로 반환됩니다.

```
double  OrderProfitCheck(
   const string        symbol,              // 기호
   ENUM_ORDER_TYPE     trade_operation,     // 주문 유형 (ORDER_TYPE_BUY 또는 ORDER_TYP
   double              volume,              // 볼륨
   double              price_open,          // 포지션 시가
   double              price_close          // 포지션 종가
   ) const
```

매개변수

Symbol

[in] 거래 작업의 기호.

trade_operation

[in] ENUM_ORDER_TYPE 열거에서의 거래 작업 유형.

볼륨

[in] 거래 작업 볼륨.

price_open

[in] 시가.

price_close

[in] 종가.

값 반환

성공하면 수익의 양 또는 오류 발생 시 EMPTY_VALUE를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

MarginCheck

거래 작업에 필요한 마진 금액을 가져오기.

```
double  MarginCheck(
   const string        symbol,              // 기호
   ENUM_ORDER_TYPE     trade_operation,     // 주문 유형 (ORDER_TYPE_BUY 또는 ORDER_TYP
   double              volume,              // 볼륨
   double              price                // 가격
   ) const
```

매개변수

Symbol

[in] 거래 작업의 기호.

trade_operation

[in] ENUM_ORDER_TYPE 열거에서의 거래 작업 유형.

볼륨

[in] 거래 작업 볼륨.

가격

[in] 거래 작업 가격.

값 반환

거래 작업에 필요한 마진 금액

© 2000-2025, MetaQuotes Ltd.

FreeMarginCheck

거래 작업 후 남은 여유 마진 금액 가져오기.

```
double  FreeMarginCheck(
   const string        symbol,              // 기호
   ENUM_ORDER_TYPE     trade_operation,     // 주문 유형 (ORDER_TYPE_BUY 또는 ORDER_TYP
   double              volume,              // 볼륨
   double              price                // 가격
   ) const
```

매개변수

Symbol

[in] 거래 작업의 기호.

trade_operation

[in] ENUM_ORDER_TYPE 열거에서의 거래 작업 유형.

볼륨

[in] 거래 작업 볼륨.

가격

[in] 거래 작업 가격.

값 반환

거래 작업 후 남은 여유 마진 금액.

© 2000-2025, MetaQuotes Ltd.

MaxLotCheck

가능한 최대 거래 작업 볼륨 가져오기.

```
double  MaxLotCheck(
   const string         symbol,              // 기호
   ENUM_ORDER_TYPE      trade_operation,     // 주문 유형 (ORDER_TYPE_BUY 또는 ORDER_TYP
   double               price,               // price
   double               percent=100          // 사용 가능한 마진 백분율 (기본값 : 100%)
   ) const
```

매개변수

Symbol

[in] 거래 작업의 기호.

trade_operation

[in] ENUM_ORDER_TYPE 열거에서의 거래 작업 유형.

가격

[in] 거래 작업 가격.

percent=100

[in] 거래 작업에 사용할 수 있는 마진의 백분율(%)입니다.

값 반환

가능한 최대 거래 작업 볼륨.

© 2000-2025, MetaQuotes Ltd.

CSymbolInfo

CSymbolInfo는 기호 속성에 쉽게 액세스할 수 있는 클래스입니다.

설명

CSymbolInfo 클래스는 기호 속성에 대한 액세스를 제공합니다.

선언

```
class CSymbolInfo : public CObject
```

제목

```
#include <Trade\SymbolInfo.mqh>
```

상속 계층

CObject
  CSymbolInfo

그룹별 클래스 메서드

Controlling	
Refresh	기호 데이터 새로 고치기
RefreshRates	기호 대기열 새로 고치기
속성	
Name	기호 이름 가져오기/설정하기
Select	"Market Watch" 기호 플래그 가져오기/설정하기
IsSynchronized	서버와의 기호 동기화를 확인하기
Volumes	
Volume	마지막 딜 볼륨 가져오기
VolumeHigh	하루의 최대 볼륨을 가져오기
VolumeLow	하루의 최소 볼륨을 가져오기
Miscellaneous	
시간	마지막 대기열 시간 가져오기
Spread	스프레드 양(점 단위) 가져오기
SpreadFloat	플로팅(floating) 스프레드의 플래그 가져오기
TicksBookDepth	틱 저장 깊이 가져오기
Levels	

© 2000-2025, MetaQuotes Ltd.

Controlling	
StopsLevel	주문의 최소 들여쓰기(점 단위) 가져오기
FreezeLevel	거래 작업 동결 차이(점 단위) 가져오기
입찰가	
Bid	현재 입찰가 가져오기
BidHigh	하루의 최대 입찰가 가져오기
BidLow	하루의 최소 입찰가 가져오기
호가	
Ask	현재 호가 가져오기
AskHigh	하루의 최대 호가 가져오기
AskLow	하루의 최소 호가 가져오기
가격	
Last	현재 마지막 가격 가져오기
LastHigh	하루의 최대 마지막 가격 가져오기
LastLow	하루의 최소 마지막 가격 가져오기
거래 모드	
TradeCalcMode	계약 비용 계산 모드 가져오기
TradeCalcModeDescription	계약 비용 계산 모드를 문자열로 가져오기
TradeMode	주문 실행 유형 가져오기
TradeModeDescription	문자열로 주문 실행 유형 가져오기
TradeExecution	거래 실행 모드 가져오기
TradeExecutionDescription	실행 모드를 문자열로 가져오기
Swaps	
SwapMode	스왑 계산 모드 가져오기
SwapModeDescription	스왑 계산 모드를 문자열로 가져오기
SwapRollover3days	삼중 스왑 청구일 가져오기
SwapRollover3daysDescription	삼중 스왑 청구일을 문자열로 가져오기
마진과 플래그	
MarginInitial	초기 마진 값 가져오기
MarginMaintenance	유지 마진 값 가져오기
MarginLong	long 포지션에 대한 마진 청구 비율을 가져오기

© 2000-2025, MetaQuotes Ltd.

Controlling	
MarginShort	short 포지션에 대한 마진 청구 비율 가져오기
MarginLimit	제한 주문에 대한 마진 청구 비율 가져오기
MarginStop	중지 주문에 대한 마진 청구 비율 가져오기
MarginStopLimit	StopLimit 주문에 대한 마진 청구 비율 가져오기
TradeTimeFlags	허용된 만료 모드의 플래그 가져오기
TradeFillFlags	허용된 채우기 모드의 플래그 가져오기
Quantization	
Digits	점 뒤의 자릿수 가져오기
Point	한 점의 값을 가져오기
TickValue	틱 값 가져오기 (가격 최소 변동)
TickValueProfit	수익성이 높은 포지션에 대해 계산된 틱 가격 가져오기
TickValueLoss	손실 포지션에 대해 계산된 틱 가격 가져오기
TickSize	가격 최소 변동 가져오기
컨트랙 크기	
ContractSize	거래 컨트랙 금액 가져오기
LotsMin	딜을 종료할 최소 볼륨 가져오기
LotsMax	딜을 종료할 최대 볼륨 가져오기
LotsStep	딜을 종료할 최소 볼륨 변경 단계 가져오기
LotsLimit	한 기호에 대해 오픈된 포지션 및 지정가 주문(방향 불감)의 최대 허용 볼륨을 가져오기
스왑 크기	
SwapLong	long 포지션 스왑 값 가져오기
SwapShort	short 포지션 스왑 값 가져오기
텍스트 속성	
CurrencyBase	기호 기준 통화 이름 가져오기
CurrencyProfit	수익 통화 이름 가져오기
CurrencyMargin	마진 통화 이름 가져오기
Bank	현재 대기열 소스 이름 가져오기
설명	기호의 문자열 설명 가져오기
Path	기호 트리의 경로 가져오기
기호 속성	

© 2000-2025, MetaQuotes Ltd.

Controlling	
SessionDeals	현재 세션의 딜 수를 가져오기
SessionBuyOrders	현재 매수 주문 수 가져오기
SessionSellOrders	현재 매도 주문 수 가져오기
SessionTurnover	현재 세션의 회전율 서머리 가져오기
SessionInterest	현재 세션의 미결제 약정의 서머리 가져오기
SessionBuyOrdersVolume	매수 주문의 현재 볼륨 가져오기
SessionSellOrdersVolume	매도 주문의 현재 볼륨 가져오기
SessionOpen	현재 세션의 시가 가져오기
SessionClose	현재 세션의 종가 가져오기
SessionAW	현재 세션의 가중 평균 가격 가져오기
SessionPriceSettlement	현재 새션의 정산 가격 가져오기
SessionPriceLimitMin	현재 세션의 최소 가격 가져오기
SessionPriceLimitMax	현재 세션의 최대 가격 가져오기
MQL5 API 기능에 액세스	
InfoInteger	지정된 정수 유형 속성 값 가져오기
InfoDouble	지정된 double 유형 속성 값 가져오기
InfoString	지정된 string 유형 속성 값 가져오기
서비스 기능	
NormalizePrice	기호 속성을 사용하여 정규화된 가격 값을 반환

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Refresh

기호 데이터 새로 고치기.

```
void  Refresh()
```

값 반환

없음.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

RefreshRates

기호 대기열 데이터 새로 고치기.

```
bool  RefreshRates()
```

값 반환

성공하면 true, 대기열을 새로 고치지 못하면 false.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Name

기호 이름 가져오기.

```
string  Name() const
```

값 반환

기호 이름.

Name

기호 이름 설정하기.

```
bool  Name(string name)
```

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

Select

"Market Watch" 기호 플래그 가져오기.

```
bool  Select() const
```

값 반환

"Market Watch" 기호 플래그.

Select

"Market Watch" 기호 플래그 설정하기.

```
bool  Select()
```

값 반환

성공하면 true, 플래그를 변경하지 못하면 false.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

IsSynchronized

서버와의 기호 동기화를 확인하기.

```
bool  IsSynchronized() const
```

값 반환

기호가 서버와 동기화가 되면 true, 그렇지 않으면 false.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Volume

마지막 딜의 볼륨 가져오기.

```
long  Volume() const
```

값 반환

마지막 딜 볼륨.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

VolumeHigh

하루의 최대 볼륨 가져오기.

```
long  VolumeHigh() const
```

값 반환

하루의 최대 볼륨.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

VolumeLow

하루의 최소 볼륨 가져오기.

```
long  VolumeLow() const
```

값 반환

하루의 최소 볼륨.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

시간

마지막 대기열 시간 가져오기.

```
datetime  Time() const
```

값 반환

마지막 대기열 시간.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Spread

스프레드 양(점 단위) 가져오기.

```
int  Spread() const
```

값 반환

스프레드 양(점 단위) 가져오기.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SpreadFloat

플로팅(floating) 스프레드의 플래그 가져오기.

```
bool  SpreadFloat() const
```

값 반환

플로팅(floating) 스프레드의 플래그.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TicksBookDepth

틱 저장 깊이 가져오기.

```
int  TicksBookDepth() const
```

값 반환

틱 저장 깊이.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

StopsLevel

주문에 대한 최소 중지 수준(점 단위)을 가져오기.

```
int  StopsLevel() const
```

값 반환

주문에 대한 최소 중지 수준(점 단위).

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

FreezeLevel

동결 수준(점 단위) 가져오기.

```
int  FreezeLevel() const
```

값 반환

동결 수준(점 단위) 차이.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Bid

현재 입찰가 가져오기.

```
double  Bid() const
```

값 반환

현재 입찰가.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

BidHigh

하루의 최대 입찰가 가져오기.

```
double  BidHigh() const
```

값 반환

하루의 최대 입찰가.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

BidLow

하루의 최소 입찰가 가져오기.

```
double  BidLow() const
```

값 반환

하루의 최소 입찰가.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Ask

현재 호가 가져오기.

```
double  Ask() const
```

값 반환

현재 호가.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

AskHigh

하루의 최대 호가 가져오기.

```
double  AskHigh() const
```

값 반환

하루의 최대 호가.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

AskLow

하루의 최소 호가 가져오기.

```
double  AskLow() const
```

값 반환

하루의 최소 호가.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Last

현재 마지막 가격 가져오기.

```
double  Last() const
```

값 반환

현재 마지막 가격 가져오기.

© 2000-2025, MetaQuotes Ltd.

LastHigh

하루의 최대 마지막 가격 가져오기.

```
double  LastHigh() const
```

값 반환

하루의 최대 마지막 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

LastLow

하루의 최소 마지막 가격 가져오기.

```
double  LastLow() const
```

값 반환

하루의 최소 마지막 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TradeCalcMode

계약 비용 계산 모드 가져오기.

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

값 반환

ENUM_SYMBOL_CALC_MODE 열거에서의 계약 비용 계산 모드.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TradeCalcModeDescription

계약 비용 계산 모드를 문자열로 가져오기.

```
string  TradeCalcModeDescription() const
```

값 반환

문자열의 계약 비용 계산 모드.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TradeMode

주문 실행 유형 가져오기.

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

값 반환

ENUM_SYMBOL_TRADE_MODE 열거에서의 주문 실행 유형.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TradeModeDescription

거래 모드를 문자열로 가져오기.

```
string  TradeModeDescription() const
```

값 반환

문자열의 거래 모드.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TradeExecution

거래 실행 모드 가져오기.

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

값 반환

ENUM_SYMBOL_TRADE_EXECUTION 열거에서의 거래 실행 모드.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TradeExecutionDescription

거래 실행 모드의 설명을 문자열로 가져오기.

```
string  TradeExecutionDescription() const
```

값 반환

문자열의 거래 실행 모드.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SwapMode

스왑 계산 모드 가져오기.

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

값 반환

ENUM_SYMBOL_SWAP_MODE 열거에서의 스왑 계산 모드.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SwapModeDescription

스왑 모드 설명을 문자열로 가져오기.

```
string  SwapModeDescription() const
```

값 반환

문자열의 스왑 모드 설명.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SwapRollover3days

스왑 롤오버 날짜 가져오기.

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

값 반환

ENUM_DAY_OF_WEEK 열거에서의 스왑 롤오버 날짜.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SwapRollover3daysDescription

스왑 롤오버 날짜를 문자열로 가져오기.

```
string  SwapRollover3daysDescription() const
```

값 반환

문자열의 스왑 롤오버 날짜.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

MarginInitial

초기 마진 값 가져오기.

```
double  MarginInitial()
```

값 반환

초기 마진 값.

참고

한 로트에서 청구된 마진(상품의 마진 통화 단위)을 반환합니다. 클라이언트가 시장에 진입할 때 고객의 지분을 확인하는 데 사용됩니다.

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

MarginMaintenance

유지 마진 값 가져오기.

```
double  MarginMaintenance()
```

값 반환

유지 마진 값.

참고

한 로트에서 청구된 마진(상품의 마진 통화 단위)을 반환합니다. 계정 상태가 변경될 때 클라이언트의 자본을 확인하는데 사용됩니다. 유지 마진이 0이면 초기 마진이 사용됩니다.

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

MarginLong

long 포지션에 대한 마진 청구 비율 가져오기.

```
double  MarginLong() const
```

값 반환

long 포지션의 마진 청구 비율.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

MarginShort

short 포지션의 마진 청구 비율 가져오기.

```
double  MarginShort() const
```

값 반환

short 포지션의 마진 청구 비율.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

MarginLimit

제한 주문의 마진 청구 비율 가져오기.

```
double  MarginLimit() const
```

값 반환

제한 주문의 마진 청구 비율

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

MarginStop

중지 주문의 마진 청구 비율 가져오기.

```
double  MarginStop() const
```

값 반환

중지 주문의 마진 청구 비율.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

MarginStopLimit

중지 제한 주문의 마진 청구 비율 가져오기.

```
double  MarginStopLimit() const
```

값 반환

중지 제한 주문 마진 청구 비율.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TradeTimeFlags

허용된 만료 모드의 플래그 가져오기.

```
int  TradeTimeFlags() const
```

값 반환

허용된 만료 모드의 플래그.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TradeFillFlags

허용된 채우기 모드의 플래그 가져오기.

```
int  TradeFillFlags() const
```

값 반환

허용된 채우기 모드의 플래그.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Digits

점 뒤의 자릿수 가져오기.

```
int  Digits() const
```

값 반환

점 뒤의 자릿수.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Point

한 점의 값 가져오기.

```
double  Point() const
```

값 반환

한 점의 값.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TickValue

틱 값 가져오기(가격 최소 변동).

```
double  TickValue() const
```

값 반환

틱 값(가격 최소 변동).

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TickValueProfit

수익성이 높은 포지션에 대해 계산된 틱 가격 가져오기.

```
double  TickValueProfit() const
```

값 반환

수익성이 높은 포지션에 대해 계산된 틱 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TickValueLoss

손실 포지션에 대해 계산된 틱 가격 가져오기.

```
double  TickValueLoss() const
```

값 반환

손실 포지션에 대해 계산된 틱 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TickSize

가격의 최소 변동 가져오기.

```
double  TickSize() const
```

값 반환

가격의 최소 변동.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

ContractSize

거래 컨트랙 금액 가져오기.

```
double  ContractSize() const
```

값 반환

거래 컨트랙 금액.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

LotsMin

딜을 종료할 최소 볼륨 가져오기.

```
double  LotsMin() const
```

값 반환

딜을 종료할 최소 볼륨.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

LotsMax

딜을 종료할 최대 볼륨 가져오기.

```
double  LotsMax() const
```

값 반환

딜을 종료할 최대 볼륨.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

LotsStep

딜을 종료할 최소 볼륨 변경 단계 가져오기.

```
double  LotsStep() const
```

값 반환

딜을 종료할 최소 볼륨 변경 단계.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

LotsLimit

한 기호에 대해 오픈된 포지션 및 지정가 주문(방향 불감)의 최대 허용 볼륨 가져오기.

```
double  LotsLimit() const
```

값 반환

한 기호에 대해 오픈된 포지션 및 지정가 주문(방향 불감)의 최대 허용 볼륨.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SwapLong

long 포지션 스왑 값 가져오기.

```
double  SwapLong() const
```

값 반환

long 포지션 스왑 값.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SwapShort

short 포지션 스왑의 값 가져오기.

```
double  SwapShort() const
```

값 반환

short 포지션 스왑 값.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

CurrencyBase

기호 기준 통화 이름 가져오기.

```
string  CurrencyBase() const
```

값 반환

기호 기준 통화 이름.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

CurrencyProfit

수익 통화 이름 가져오기.

```
string  CurrencyProfit() const
```

값 반환

수익 통화 이름.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

CurrencyMargin

마진 통화 이름 가져오기.

```
string  CurrencyMargin() const
```

값 반환

마진 통화 이름.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Bank

현재 대기열 소스의 이름 가져오기.

```
string  Bank() const
```

값 반환

현재 대기열 소스의 이름.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

설명

기호의 문자열 설명 가져오기.

```
string  Description() const
```

값 반환

기호의 문자열 설명.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Path

기호 트리의 경로 가져오기.

```
string  Path() const
```

값 반환

기호 트리의 경로 가져오기.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionDeals

현재 세션의 딜 수 가져오기.

```
long  SessionDeals() const
```

값 반환

현재 세션의 딜 수.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionBuyOrders

현재 매수 주문 수 가져오기.

```
long  SessionBuyOrders() const
```

값 반환

현재 매수 주문 수.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionSellOrders

현재 매도 주문 수 가져오기.

```
long  SessionSellOrders() const
```

값 반환

현재 매도 주문 수.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionTurnover

현재 세션의 회전율 서머리 가져오기.

```
double  SessionTurnover() const
```

값 반환

현재 세션의 회전율 서머리.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionInterest

현재 세션의 미결제 약정 서머리 가져오기.

```
double  SessionInterest() const
```

값 반환

현재 세션의 미결제 약정의 서머리.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionBuyOrdersVolume

매수 주문 현재 볼륨 가져오기.

```
double  SessionBuyOrdersVolume() const
```

값 반환

현재 매수 주문 볼륨.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionSellOrdersVolume

매도 주문 현재 볼륨 가져오기.

```
double  SessionSellOrdersVolume() const
```

값 반환

현재 매도 주문 볼륨.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionOpen

현재 세션의 시가 가져오기.

```
double  SessionOpen() const
```

값 반환

현재 새션의 시가.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionClose

현재 세션의 종가 가져오기.

```
double  SessionClose() const
```

값 반환

현재 세션의 종가.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionAW

현재 세션의 가중 평균 가격 가져오기.

```
double  SessionAW() const
```

값 반환

현재 세션의 평균 가중 평균 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionPriceSettlement

현재 세션의 정산 가격 가져오기.

```
double  SessionPriceSettlement() const
```

값 반환

현재 세션의 정산 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionPriceLimitMin

현재 세션의 최소 가격 가져오기.

```
double  SessionPriceLimitMin() const
```

값 반환

현재 세션의 최소 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

SessionPriceLimitMax

현재 세션의 최대 가격 가져오기.

```
double  SessionPriceLimitMax() const
```

값 반환

현재 세션의 최대 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

지정한 정수 유형 속성 값 가져오기.

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,      // 속성 ID
   long&                     var           // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_SYMBOL_INFO_INTEGER 열거의 정수 유형 속성 ID.

var

[out] 결과를 배치할 long 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

지정된 double 유형 속성 값을 가져오기.

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,      // 속성 ID
   double&                  var           // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_SYMBOL_INFO_DOUBLE 열거의 double 유형 속성 ID.

var

[out] 결과를 배치할 double 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoString

지정한 문자열 유형 속성의 값을 가져오기.

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,      // 속성 ID
   string&                  var           // 변수 참조
   ) const
```

매개변수

prop_id

[in] 텍스트 속성의 ID.

var

[out] 결과를 배치할 string 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

NormalizePrice

기호 속성을 사용하여 정규화된 가격을 반환.

```
double  NormalizePrice(
   double  price   // 가격
   ) const
```

매개변수

가격

[in] 가격.

값 반환

정규화된 가격.

참고

기호는 Name 메서드로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

COrderInfo

COrderInfo는 지정가 주문 속성에 쉽게 액세스할 수 있는 클래스.

설명

COrderInfo 클래스는 지정가 주문 속성에 대한 액세스를 제공합니다.

선언

```
class COrderInfo : public CObject
```

제목

```
#include <Trade\OrderInfo.mqh>
```

상속 계층

CObject
  COrderInfo

그룹별 클래스 메서드

정수 유형 속성에 대한 액세스	
Ticket	액세스하기 위해 이전에 선택한 주문의 티켓 가져오기
TimeSetup	주문 발주 시간을 가져오기
TimeSetupMsc	주문 발주 시간(밀리초) 수신: 1970년 1월 1일 이후
OrderType	주문 유형 가져오기
OrderTypeDescription	주문 유형을 문자열로 가져오기
State	주문 상태 가져오기
StateDescription	주문 상태를 문자열로 가져오기
TimeExpiration	주문 만료 시간 가져오기
TimeDone	주문 실행 또는 취소 시간을 가져오기
TimeDoneMsc	주문 실행 또는 취소 시간(밀리초): 1970년 1월 1일 이후
TypeFilling	나머지를 기준으로 주문 실행 유형 가져오기
TypeFillingDescription	문자열로 나머지를 기준으로 주문 실행 유형을 가져오기
TypeTime	만료 시점의 주문 유형을 가져오기
TypeTimeDescription	문자열로 만료 시간을 기준으로 주문 유형 가져오기
Magic	주문을 수행한 엑스퍼트의 ID 가져오기
PositionId	포지션 ID 가져오기

© 2000-2025, MetaQuotes Ltd.

정수 유형 속성에 대한 액세스	
double 유형 속성에 액세스	
VolumeInitial	주문의 초기 볼륨 가져오기
VolumeCurrent	채워지지 않은 주문 볼륨 가져오기
PriceOpen	주문 가격 가져오기
StopLoss	주문의 손절매 가져오기
TakeProfit	주문의 차익 실현 가져오기
PriceCurrent	주문 기호별 현재 가격을 가져오기
PriceStopLimit	제한 주문의 가격을 가져오기
텍스트 설정에 액세스	
Symbol	주문 기호 이름 가져오기
Comment	주문 설명 가져오기
MQL5 API 기능에 액세스	
InfoInteger	지정된 정수 유형 속성 값 가져오기
InfoDouble	지정된 double 유형 속성 값 가져오기
InfoString	지정된 문자열 유형 속성 값 가져오기
State	
StoreState	주문 매개변수 저장하기
CheckState	저장된 매개변수와 현재 매개별수를 확인하기
선택	
Select	속성에 대한 추가 액세스를 위해 티켓별로 주문을 선택하기
SelectByIndex	속성에 대한 추가 액세스를 위해 인덱스별로 주문을 선택

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

티켓

주문의 티켓 가져오기.

```
ulong  Ticket() const
```

값 반환

성공하면 주문 티켓을, 그렇지 않으면 ULONG_MAX.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeSetup

주문 발주 시간 가져오기.

```
datetime  TimeSetup() const
```

값 반환

주문 발주 시간.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeSetupMsc

시간(밀리초) 수신: 1970년 1월 1일 이후, 실행을 위해 주문을 발주한 시간.

```
ulong  TimeSetupMsc() const
```

값 반환

발주 시간: 1970년 1월 1일 이후 실행을 위한 주문.

참고

순서는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드로 예비적으로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

OrderType

주문 유형 가져오기.

```
ENUM_ORDER_TYPE  OrderType()
```

값 반환

ENUM_ORDER_TYPE 열거의 주문 유형.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeDescription

주문 유형을 문자열로 가져오기.

```
string  TypeDescription() const
```

값 반환

문자열 주문 유형.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

State

주문 상태 가져오기.

```
ENUM_ORDER_STATE  State() const
```

값 반환

ENUM_ORDER_STATE 열거 순서 상태.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

StateDescription

주문 상태를 문자열로 가져오기.

```
string  StateDescription() const
```

값 반환

문자열 주문 상태.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeExpiration

주문 만료 시간 가져오기.

```
datetime  TimeExpiration() const
```

값 반환

발주에 설정된 주문 만료 시간.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeDone

주문 실행 또는 취소 시간 가져오기.

```
datetime  TimeDone() const
```

값 반환

주문 실행 또는 취소 시간.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeDoneMsc

주문 실행 또는 취소 시간(밀리초) 수신: 1970년 1월 1일 이후.

```
ulong  TimeDoneMsc() const
```

값 반환

주문 실행 또는 취소 시간(밀리초) : 1970년 1월 1일 이후.

참고

순서는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드로 예비적으로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeFilling

주문 채우기 유형 가져오기.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

값 반환

ENUM_ORDER_TYPE_FILLING 열거의 주문 채우기 유형.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeFillingDescription

문자열로 주문 채우기 유형 가져오기.

```
string  TypeFillingDescription() const
```

값 반환

문자열 주문 채우기 유형.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeTime

만료 시점의 주문 유형 가져오기.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

값 반환

ENUM_ORDER_TYPE_TIME 열거에서 만료 시점의 주문 유형.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeTimeDescription

문자열로 만료 시간을 기준으로 주문 유형을 가져오기.

```
string  TypeTimeDescription() const
```

값 반환

문자열로 만료 시간을 기준으로 주문 유형.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Magic

주문을 발주한 엑스퍼트 어드바이저 ID 가져오기.

```
long  Magic() const
```

값 반환

주문을 발주한 엑스퍼트 어드바이저의 ID.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PositionId

포지션의 ID 가져오기.

```
long  PositionId() const
```

값 반환

주문이 관여된 포지션의 ID.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

VolumeInitial

주문 초기 볼륨 가져오기.

```
double  VolumeInitial() const
```

값 반환

주문 초기 볼륨.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

VolumeCurrent

채워지지 않은 주문 볼륨 가져오기.

```
double  VolumeCurrent() const
```

값 반환

채워지지 않은 주문 볼륨.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PriceOpen

주문 가격 가져오기.

```
double  PriceOpen() const
```

값 반환

주문 발주 가격.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

StopLoss

주문의 손절매 가져오기.

```
double  StopLoss() const
```

값 반환

주문의 손절매.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TakeProfit

주문의 차익 실현 가져오기.

```
double  TakeProfit() const
```

값 반환

주문의 차익 실현.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

주문 기호별 현재 가격 가져오기.

```
double  PriceCurrent() const
```

값 반환

주문 기호별 현재 가격.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PriceStopLimit

지정가 주문 가격 가져오기.

```
double  PriceStopLimit() const
```

값 반환

지정가 주문 가격.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Symbol

주문 기호의 이름 가져오기.

```
string  Symbol() const
```

값 반환

주문 기호 이름.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Comment

주문 설명 가져오기.

```
string  Comment() const
```

값 반환

주문 설명.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

지정한 정수 유형 속성 값 가져오기.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,     // 속성 ID
   long&                        var          // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_ORDER_PROPERTY_INTEGER 열거의 정수 유형 속성 ID.

var

[out] 결과를 배치할 long 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

지정된 double 유형 속성 값을 가져오기.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // 속성 ID
   double&                     var          // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_ORDER_PROPERTY_DOUBLE 열거의 double 유형 속성 ID.

var

[out] 결과를 배치할 double 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoString

지정한 문자열 유형 속성의 값을 가져오기.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,     // 속성 ID
   string&                     var          // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_ORDER_PROPERTY_STRING 열거의 문자열 속성 ID.

var

[out] 결과를 배치할 string 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

주문은 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

StoreState

주문 매개변수 저장하기.

```
void  StoreState()
```

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

CheckState

저장된 매개변수와 비교하여 현재 매개변수를 확인하기.

```
bool  CheckState()
```

값 반환

true - StoreState() 메서드의 마지막 호출 이후 주문 매개변수가 변경된 경우, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Select

속성에 대한 추가 액세스를 위해 티켓별로 주문을 선택하기.

```
bool  Select(
   ulong  ticket   // 주문 티켓
  \)
```

값 반환

성공하면 true, 순서를 선택하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

인덱스별 주문 선택하기.

```
bool  SelectByIndex(
   int  index   // 주문 인덱스
  \)
```

매개변수

인덱스

[in] 주문 인덱스.

값 반환

성공하면 true, 순서를 선택하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

CHistoryOrderInfo

CHistoryOrderInfo는 내역 순서 속성에 쉽게 액세스하기 위한 클래스입니다.

설명

CHistoryOrderInfo 클래스는 내역 순서 속성에 쉽게 액세스할 수 있도록 합니다.

선언

```
class CHistoryOrderInfo : public CObject
```

제목

```
#include <Trade\HistoryOrderInfo.mqh>
```

상속 계층

CObject

CHistoryOrderInfo

그룹별 클래스 메서드

정수 유형 속성에 대한 액세스	
TimeSetup	주문 발주 시간을 가져오기
TimeSetupMsc	주문 발주 시간(밀리초) 수신: 1970년 1월 1일 이후
OrderType	주문 유형 가져오기
OrderTypeDescription	주문 유형을 문자열로 가져오기
State	주문 상태 가져오기
StateDescription	주문 상태를 문자열로 가져오기
TimeExpiration	주문 만료 시간 가져오기
TimeDone	주문 실행 또는 취소 시간을 가져오기
TimeDoneMsc	주문 실행 또는 취소 시간(밀리초): 1970년 1월 1일 이후
TypeFilling	나머지를 기준으로 주문 실행 유형 가져오기
TypeFillingDescription	문자열로 나머지를 기준으로 주문 실행 유형을 가져오기
TypeTime	만료 시점의 주문 유형을 가져오기
TypeTimeDescription	문자열로 만료 시간을 기준으로 주문 유형 가져오기
Magic	주문을 수행한 엑스퍼트의 ID 가져오기
PositionId	포지션 ID 가져오기
double 유형 속성에 액세스	

© 2000-2025, MetaQuotes Ltd.

정수 유형 속성에 대한 액세스	
VolumeInitial	주문의 초기 볼륨 가져오기
VolumeCurrent	채워지지 않은 주문 볼륨 가져오기
PriceOpen	주문 가격 가져오기
StopLoss	주문의 손절매 가져오기
TakeProfit	주문의 차익 실현 가져오기
PriceCurrent	주문 기호별 현재 가격을 가져오기
PriceStopLimit	제한 주문의 가격을 가져오기
텍스트 설정에 액세스	
Symbol	주문 기호 가져오기
Comment	주문 설명 가져오기
MQL5 API 기능에 액세스	
InfoInteger	지정된 정수 유형 속성 값 가져오기
InfoDouble	지정된 double 유형 속성 값 가져오기
InfoString	지정된 문자열 유형 속성 값 가져오기
선택	
티켓	티켓을 가져오기/주문 선택하기
SelectByIndex	인덱스별 순서 선택하기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

TimeSetup

주문 발주 시간 가져오기.

```
datetime  TimeSetup() const
```

값 반환

주문 발주 시간.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeSetupMsc

실행 주문 발주 시간(밀리초) 수신: 1970년 1월 1일 이후.

```
ulong  TimeSetupMsc() const
```

값 반환

실행 주문 발주 시간(밀리초) : 1970년 1월 1일.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 액세스 하기 위해 미리 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

OrderType

주문 유형 가져오기.

```
ENUM_ORDER_TYPE  OrderType() const
```

값 반환

ENUM_ORDER_TYPE 열거의 주문 유형.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeDescription

주문 유형을 문자열로 가져오기.

```
string  TypeDescription() const
```

값 반환

문자열 주문 유형.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

State

주문 상태 가져오기.

```
ENUM_ORDER_STATE  State() const
```

값 반환

ENUM_ORDER_STATE 열거 순서 상태.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

StateDescription

주문 상태를 문자열로 가져오기.

```
string  StateDescription() const
```

값 반환

문자열 주문 상태.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeExpiration

주문 만료 시간 가져오기.

```
datetime  TimeExpiration() const
```

값 반환

발주에 설정된 주문 만료 시간.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeDone

주문 실행 또는 취소 시간 가져오기.

```
datetime  TimeDone() const
```

값 반환

주문 실행 또는 취소 시간.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeDoneMsc

주문 실행 또는 취소 시간(밀리초) 수신: 1970년 1월 1일 이후.

```
ulong  TimeDoneMsc() const
```

값 반환

주문 실행 또는 취소 시간(밀리초) : 1970년 1월 1일 이후.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 액세스 하기 위해 미리 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeFilling

나머지를 기준으로 주문 실행 유형을 가져오기.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

값 반환

ENUM_ORDER_TYPE_FILLING 열거형의 나머지에 따른 주문 실행 유형.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeFillingDescription

문자열로 나머지를 기준으로 주문 실행 유형을 가져오기.

```
string  TypeFillingDescription() const
```

값 반환

나머지를 문자열로 실행 유형 주문.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeTime

만료 시점의 주문 유형 가져오기.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

값 반환

ENUM_ORDER_TYPE_TIME 열거에서 만료 시점의 주문 유형.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeTimeDescription

문자열로 만료 시간을 기준으로 주문 유형을 가져오기.

```
string  TypeTimeDescription() const
```

값 반환

문자열로 만료 시간을 기준으로 주문 유형.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Magic

주문을 발주한 엑스퍼트 어드바이저 ID 가져오기.

```
long  Magic() const
```

값 반환

주문을 발주한 엑스퍼트 어드바이저의 ID.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PositionId

포지션의 ID 가져오기.

```
long  PositionId() const
```

값 반환

주문이 관여된 포지션의 ID.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

VolumeInitial

주문 초기 볼륨 가져오기.

```
double  VolumeInitial() const
```

값 반환

주문 초기 볼륨.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

VolumeCurrent

채워지지 않은 주문 볼륨 가져오기.

```
double  VolumeCurrent() const
```

값 반환

채워지지 않은 주문 볼륨.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PriceOpen

주문 가격 가져오기.

```
double  PriceOpen() const
```

값 반환

주문 발주 가격.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

StopLoss

주문의 손절매 가격 가져오기.

```
double  StopLoss() const
```

값 반환

주문의 손절매 가격.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TakeProfit

주문 이익 실현 가격을 가져오기.

```
double  TakeProfit() const
```

값 반환

주문 이익 실현 가격.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

주문 기호의 현재 가격을 가져오기.

```
double  PriceCurrent() const
```

값 반환

주문 기호 현재 가격.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PriceStopLimit

지정가 주문 가격 가져오기.

```
double  PriceStopLimit() const
```

값 반환

지정가 주문 가격.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Symbol

주문 기호의 이름 가져오기.

```
string  Symbol() const
```

값 반환

주문 기호 이름.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Comment

주문 설명 가져오기.

```
string  Comment() const
```

값 반환

주문 설명.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

지정한 정수 유형 속성 값 가져오기.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,     // 속성 ID
   long&                        var          // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_ORDER_PROPERTY_INTEGER 열거의 정수 유형 속성 ID.

var

[out] 결과를 배치할 long 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

지정된 double 유형 속성 값을 가져오기.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // 속성 ID
   double&                     var          // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_ORDER_PROPERTY_DOUBLE 열거의 double 유형 속성 ID.

var

[out] 결과를 배치할 double 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoString

지정한 문자열 유형 속성의 값을 가져오기.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // 속성 ID
   string&                     var           // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_ORDER_PROPERTY_STRING 열거의 텍스트 속성 ID.

var

[out] 결과를 배치할 string 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

내역 순서는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방법을 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Ticket (Get method)

주문 티켓 가져오기.

```
ulong  Ticket() const
```

값 반환

주문 티켓.

Ticket (Set method)

추가 작업의 순서를 선택하기.

```
void  Ticket(
   ulong  ticket      // 티켓
  \)
```

매개변수

티켓

[in] 주문 티켓.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

인덱스별 내역 순서 선택하기.

```
bool  SelectByIndex(
   int  index   // 주문 인덱스
  \)
```

매개변수

인덱스

[in] 내역 순서 인덱스.

값 반환

성공하면 true, 순서를 선택하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

CPositionInfo

CPositionInfo는 오픈 포지션 속성에 쉽게 액세스할 수 있는 클래스입니다.

설명

CPositionInfo 클래스는 오픈 포지션 속성에 쉽게 액세스할 수 있도록 합니다.

선언

```
class CPositionInfo : public CObject
```

제목

```
#include <Trade\PositionInfo.mqh>
```

상속 계층

CObject

CPositionInfo

그룹별 클래스 메서드

정수 유형 속성에 대한 액세스	
Time	포지션 오프닝 시간 가져오기
TimeMsc	시간(밀리초) 수신: 1970년 1월 1일 이후 포지션 오프닝
TimeUpdate	수신: 1970년 1월 1일 이후 포지션 변경 시간(초)
TimeUpdateMsc	수신: 1970년 1월 1일 이후 포지션 변경 시간(밀리초)
PositionType	포지션 유형 가져오기
TypeDescription	문자열로 포지션 유형 가져오기
Magic	오픈된 포지션의 엑스퍼트 ID 가져오기
식별자	포지션 ID 가져오기
double 유형 속성에 액세스	
Volume	포지션 볼륨 가져오기
PriceOpen	포지션 오프닝 가격 가져오기
StopLoss	포지션의 손절매 가격 가져오기
TakeProfit	포지션의 차익 실현 가져오기
PriceCurrent	포지션 기호별 현재 가격 가져오기
Commission	포지션별 커미션 금액 가져오기
Swap	포지션별 스왑 금액 가져오기

© 2000-2025, MetaQuotes Ltd.

정수 유형 속성에 대한 액세스	
Profit	포지션별 경상 이익 금액 가져오기
텍스트 설정에 액세스	
Symbol	포지션 기호의 이름 가져오기
Comment	포지션의 설명 가져오기
MQL5 API 기능에 액세스	
InfoInteger	지정된 정수 유형 속성 값 가져오기
InfoDouble	지정된 double 유형 속성 값 가져오기
InfoString	지정된 string 유형 속성 값 가져오기
선택	
Select	포지션 선택하기
SelectByIndex	인덱스별 포지션 선택하기
SelectByMagic	지정된 기호 이름 및 매직 넘버를 가진 포지션 선택하기
SelectByTicket	티켓별 포지션 선택하기
State	
StoreState	포지션 매개변수 저장하기
CheckState	저장된 매개변수와 현재 매개별수를 확인하기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

시간

포지션 오프닝 시간 가져오기.

```
datetime  Time() const
```

값 반환

포지션 오프닝 시간.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeMsc

포지션 오프닝 시간(밀리초) 수신: 1970년 1월 1일 이후.

```
ulong  TimeMsc() const
```

값 반환

포지션 오프닝 시간(밀리초) : 1970년 1월 1일 이후.

참고

위치는 선택 (기호별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 액세스하기 위해 미리 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeUpdate

시간 수신: 1970년 1월 1일 이후 포지션 변경 시간(초).

```
datetime  TimeUpdate() const
```

값 반환

시간(초): 1970년 1월 1일 이후 포지션 변경.

참고

위치는 선택 (기호별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 액세스하기 위해 미리 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeUpdateMsc

시간 수신: 1970년 1월 1일 이후 포지션 변경 시간(밀리초).

```
ulong  TimeUpdateMsc() const
```

값 반환

시간(밀리초): 1970년 1월 1일 이후 포지션 변경.

참고

위치는 선택 (기호별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 액세스하기 위해 미리 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PositionType

포지션 유형 가져오기.

```
ENUM_POSITION_TYPE  PositionType() const
```

값 반환

ENUM_POSITION_TYPE 열거의 포지션 유형.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeDescription

포지션 유형을 문자열로 가져오기.

```
string  TypeDescription() const
```

값 반환

문자열 포지션 유형.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Magic

포지션을 오픈한 엑스퍼트 어드바이저 ID 가져오기

```
long  Magic() const
```

값 반환

포지션을 오픈한 엑스퍼트 어드바이저 ID.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

식별자

포지션의 ID 가져오기.

```
long  Identifier() const
```

값 반환

포지션의 ID.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Volume

포지션 볼륨 가져오기.

```
double  Volume() const
```

값 반환

포지션의 볼륨.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PriceOpen

포지션 오프닝 가격 가져오기.

```
double  PriceOpen() const
```

값 반환

포지션 시가.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

StopLoss

포지션의 손절매 가격을 가져오기.

```
double  StopLoss() const
```

값 반환

포지션의 손절매 가격.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TakeProfit

포지션 차익 실현 가격 가져오기.

```
double  TakeProfit() const
```

값 반환

포지션의 차익 실현 가격 .

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PriceCurrent

포지션 기호별 현재 가격 가져오기.

```
double  PriceCurrent() const
```

값 반환

포지션 기호별 현재 가격.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Commission

포지션의 커미션 금액 가져오기.

```
double  Commission() const
```

값 반환

포지션의 커미션 금액(예금 통화).

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Swap

포지션의 스왑 금액 가져오기.

```
double  Swap() const
```

값 반환

포지션의 스왑 금액(예금 통화).

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Profit

포지션의 경상 이익 금액 가져오기.

```
double  Profit() const
```

값 반환

포지션의 경상 이익 금액(예금 통화).

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Symbol

포지션 기호의 이름 가져오기.

```
string  Symbol() const
```

값 반환

포지션 기호 이름.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Comment

포지션 설명 가져오기.

```
string  Comment() const
```

값 반환

포지션 설명.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

지정한 정수 유형 속성 값 가져오기.

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,      // 속성 ID
   long&                           var           // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_POSITION_PROPERTY_INTEGER 열거의 정수 유형 속성 ID.

var

[out] 결과를 배치할 long 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

지정된 double 유형 속성 값을 가져오기.

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // 속성 ID
   double&                        var          // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_POSITION_PROPERTY_DOUBLE 열거의 double 유형 속성 ID.

var

[in] 결과를 배치할 double 유형 변수.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoString

지정한 문자열 유형 속성의 값을 가져오기.

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // 속성 ID
   string&                        var           // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_POSITION_PROPERTY_STRING 열거의 텍스트 유형 속성 ID.

var

[out] 결과를 배치할 string 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

위치는 선택 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Select

추가 작업을 위한 포지션 선택.

```
bool  Select(
   const string  symbol      // 기호
  \)
```

매개변수

Symbol

[in] 포지션 선택 기호.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

속성에 대한 추가 액세스를 위해 인덱스별로 위치를 선택하기.

```
bool  SelectByIndex(
   int  index   // 포지션 인덱스
   );
```

값 반환

성공하면 true, 포지션을 선택하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

SelectByMagic

금융상품의 이름과 추가 작업에 사용할 매직 넘버를 기준으로 위치를 선택하기.

```
bool  SelectByMagic(
   const string  symbol, // 기호 이름
   const ulong   magic   // 매직 넘버
   );
```

매개변수

Symbol

[in] 기호 이름.

magic

[in] 포지션의 매직 넘버.

값 반환

성공하면 true, 포지션을 선택하지 못하면 false.

© 2000-2025, MetaQuotes Ltd.

SelectByTicket

추가 작업을 위해 티켓별로 위치를 선택하기.

```
bool  SelectByTicket(
   ulong  ticket      // 포지션 티켓
   );
```

매개변수

티켓

[in] 포지션 티켓.

값 반환

성공하면 true, 선택된 포지션이 없으면 false.

© 2000-2025, MetaQuotes Ltd.

StoreState

포지션 매개변수를 저장.

```
void  StoreState()
```

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

CheckState

저장된 매개변수와 비교하여 현재 매개변수를 확인하기.

```
bool  CheckState()
```

값 반환

true - StoreState() 메서드의 마지막 호출 이후 포지션 매개변수가 변경된 경우, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CDealInfo

CDealInfo는 거래 속성에 쉽게 액세스하기 위한 클래스입니다.

설명

CDealInfo 클래스는 딜 속성에 대한 액세스를 제공합니다.

선언

```
class CDealInfo : public CObject
```

제목

```
#include <Trade\DealInfo.mqh>
```

상속 계층

CObject
  CDealInfo

그룹별 클래스 메서드

정수 유형 속성에 대한 액세스	
Order	딜 실행 순서 가져오기
Time	딜 실행 시간 가져오기
TimeMsc	딜 실행 시간(밀리초) 수신: 1970년 1월 1일 이후
DealType	딜 유형 가져오기
TypeDescription	딜 유형을 문자열로 가져오기
Entry	딜 방향을 가져오기
EntryDescription	딜 방향을 문자열로 가져오기
Magic	거래를 실행한 엑스퍼트의 ID 가져오기
PositionId	딜과 관여된 포지션의 ID 가져오기
double 유형 속성에 액세스	
Volume	딜 볼륨 가져오기
Price	딜 가격 가져오기
Commision	딜의 커미션 금액을 가져오기
Swap	포지션이 닫힐 때의 스왑 금액 가져오기
Profit	딜의 재무 결과 가져오기
텍스트 설정에 액세스	

© 2000-2025, MetaQuotes Ltd.

정수 유형 속성에 대한 액세스	
Symbol	딜 기호 이름 가져오기
Comment	딜 설명 가져오기
MQL5 API 기능에 액세스	
InfoInteger	지정된 정수 유형 속성 값 가져오기
InfoDouble	지정된 double 유형 속성 값 가져오기
InfoString	지정된 문자열 유형 속성 값 가져오기
선택	
Ticket	티켓 가져오기/딜 선택하기
SelectByIndex	인덱스별로 딜 선택하기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Order

딜이 실행되는 순서를 가져오기.

```
long  Order() const
```

값 반환

딜이 실행된 순서.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메서드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

시간

딜 실행 시간 가져오기.

```
datetime  Time() const
```

값 반환

딜 실행 시간.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TimeMsc

1970년 1월 1일 이후 딜 실행 시간(밀리초)을 수신.

```
ulong  TimeMsc() const
```

값 반환

딜 실행 시간(밀리초) 수신: 1970년 1월 1일 이후.

참고

딜은 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 방식으로 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

DealType

딜 유형 가져오기.

```
ENUM_DEAL_TYPE  DealType() const
```

값 반환

ENUM_DEAL_TYPE 열거의 딜 유형.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

TypeDescription

딜 유형을 문자열로 가져오기.

```
string  TypeDescription() const
```

값 반환

문자열 딜 유형.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Entry

딜 방향 비교하기.

```
ENUM_DEAL_ENTRY  Entry() const
```

값 반환

딜 방향 (ENUM_DEAL_ENTRY 열거 값).

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

EntryDescription

딜 방향을 문자열로 가져오기.

```
string  EntryDescription() const
```

값 반환

딜 방향 문자열.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Magic

딜을 실행한 엑스퍼트 어드바이저의 ID 가져오기.

```
long  Magic() const
```

값 반환

딜을 실행한 엑스퍼트 어드바이저의 ID.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PositionId

거래가 관련된 위치 ID를 가져오기.

```
long  PositionId() const
```

값 반환

거래가 여된 포지션 ID.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Volume

딜 볼륨 가져오기.

```
double  Volume() const
```

값 반환

딜 볼륨.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

가격

딜 가격 가져오기.

```
double  Price() const
```

값 반환

딜 가격.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Commission

딜의 커미션 금액 가져오기.

```
double  Commission() const
```

값 반환

딜 커미션의 금액.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Swap

포지션이 닫힐 때의 스왑 금액을 가져오기.

```
double  Swap() const
```

값 반환

포지션의 닫혔을 때의 스왑 금액.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Profit

딜의 재무 결과를 가져오기.

```
double  Profit() const
```

값 반환

딜의 재무 결과 (예금 통화).

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Symbol

딜 기호 이름 가져오기.

```
string  Symbol() const
```

값 반환

딜 기호 이름.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Comment

딜 설명 가져오기.

```
string  Comment() const
```

값 반환

딜 설명.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoInteger

지정한 정수 유형 속성 값 가져오기.

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,     // 속성 ID
   long&                       var          // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_DEAL_PROPERTY_INTEGER 열거의 정수 유형 속성 ID.

var

[out] 결과를 배치할 long 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoDouble

지정된 double 유형 속성 값을 가져오기.

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // 속성 ID
   double&                    var           // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_DEAL_PROPERTY_DOUBLE 열거의 double 유형 속성 ID.

var

[out] 결과를 배치할 double 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

InfoString

지정한 문자열 유형 속성의 값을 가져오기.

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // 속성 ID
   string&                    var           // 변수 참조
   ) const
```

매개변수

prop_id

[in] ENUM_DEAL_PROPERTY_STRING 열거의 텍스트 속성 ID.

var

[out] 결과를 배치할 string 유형 변수에 대한 참조.

값 반환

성공하면 true, 속성 값을 가져오지 못하면 false.

참고

거래는 티켓 (티켓별) 또는 SelectByIndex (인덱스별) 메스드를 사용하여 선택해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Ticket (Get method)

딜 티켓 가져오기.

```
ulong  Ticket() const
```

값 반환

딜 티켓.

Ticket (Set method)

추가 작업을 위한 포지션 선택.

```
void  Ticket(
   ulong  ticket      // 티켓
  \)
```

매개변수

티켓

[in] 딜 티켓.

© 2000-2025, MetaQuotes Ltd.

SelectByIndex

속성에 대한 추가 액세스를 위해 인덱스별로 딜을 선택하기.

```
bool  SelectByIndex(
   int  index   // 주문 인덱스
  \)
```

값 반환

성공하면 true, 딜을 선택할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

CTrade

CTrade 거래 기능에 쉽게 접근할 수 있는 클래스입니다.

설명

CTrade 클래스는 거래 기능에 쉽게 액세스할 수 있도록 합니다.

선언

```
class CTrade : public CObject
```

제목

```
#include <Trade\Trade.mqh>
```

상속 계층

CObject

CTrade

직계 후손

CExpertTrade

그룹별 클래스 메서드

매개변수 설정	
LogLevel	로깅 수준 설정하기
SetExpertMagicNumber	엑스퍼트 ID 설정하기
SetDeviationInPoints	허용 편차 설정하기
SetTypeFilling	주문 채우기 유형 설정하기
SetTypeFillingBySymbol	지정된 기호 설정에 따라 주무느이 채우기 유형을 설정하기
SetAsyncMode	거래 작업에 대한 비동기 모드를 설정하기
SetMarginMode	현재 계정 설정에 따라 마진 계산 모드를 설정하기
주문 작업	
OrderOpen	지정된 매개변수를 사용하여 지정가 주문을 발주하기
OrderModify	지정가 주문 매개변수 수정
OrderDelete	지정가 주문 삭제하기
포지션 작업	
PositionOpen	지정된 매개변수가 있는 포지션 열기
PositionModify	지정된 기호 또는 포지션 티켓으로 포지션 매개변수 수정하기

© 2000-2025, MetaQuotes Ltd.

매개변수 설정	
PositionClose	지정된 기호의 포지션 닫기
PositionClosePartial	지정된 기호 또는 지정된 티켓의 포지션을 부분적으로 닫기
PositionCloseBy	지정된 티켓과 반대 포지션으로 포지션을 닫기
추가 메서드	
Buy	지정된 매개변수를 사용하여 long 포지션 열기
Sell	지정된 매개변수를 사용하여 short 포지션 열기
BuyLimit	지정된 매개변수를 사용하여 매수 제한 유형의 지정가 주문을 발주하기
BuyStop	지정된 매개변수를 사용하여 매수 중지 유형의 지정가 주문을 발주하기
SellLimit	지정된 매개변수를 사용하여 매도 제한 유형의 지정가 주문을 발주하기
SellStop	지정된 매개변수를 사용하여 매도 중지 유형의 지정가 주문을 발주하기
마지막 요청 매개변수에 대한 액세스	
Request	마지막 요청 구조의 복사본을 가져오기
RequestAction	거래 작업 유형 가져오기
RequestActionDescription	거래 작업 유형을 문자열로 가져오기
RequestMagic	엑스퍼트 어드바이저의 매직 넘버 가져오기
RequestOrder	마지막 요청에 사용된 주문 티켓을 가져오기
RequestSymbol	마지막 요청에 사용된 기호 이름을 가져오기
RequestVolume	마지막 요청에 사용된 거래 볼륨을 가져오기(로트 단위)
RequestPrice	마지막 요청에 사용된 가격을 가져오기
RequestStopLimit	마지막에 요청된 중지 제한 유형의 지정가 주문 가격을 가져오기
RequestSL	마지막 요청에 사용된 주문의 손절매 가격을 가져오기
RequestTP	마지막 요청에 사용된 주문의 이익 실현 가격 가져오기
RequestDeviation	마지막 요청에 사용된 주문의 최대 허용 가격 편차를 가져오기
RequestType	마지막 요청에 사용된 주문 유형 가져오기
RequestTypeDescription	마지막 요청에 사용된 주문의 유형(문자열로)을 가져오기
RequestTypeFilling	마지막 요청에 사용된 주문의 채우기 유형 가져오기
RequestTypeFillingDescription	마지막 요청에 사용된 주문의 채우기 유형(문자열로)을 가져오기

© 2000-2025, MetaQuotes Ltd.

매개변수 설정	
RequestTypeTime	마지막 요청에 사용된 주문의 유효 기간을 가져오기
RequestTypeTimeDescription	마지막 요청에 사용된 주문의 유효 기간(문자열로)을 가져오기
RequestExpiration	마지막 요청에 사용된 주문의 만료 시간을 가져오기
RequestComment	마지막 요청에 사용된 주문의 설명을 가져오기
RequestPosition	포지션 티켓 가져오기
RequestPositionBy	반대 포지션 티켓 가져오기
마지막 요청 확인 결과에 대한 액세스	
CheckResult	마지막 요청 검사 결과의 구조 복사본 가져오기.
CheckResultRetcode	요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 retcode 필드 값 가져오기
CheckResultRetcodeDescription	요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 retcode 필드에 대한 문자열 설명 가져오기
CheckResultBalance	요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 밸런스 필드 값을 가져오기
CheckResultEquity	요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 자기자본 필드 값을 가져오기
CheckResultProfit	거래 작업을 실행한 후 플로팅(floating) 이익의 값을 가져오기.
CheckResultMargin	요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 여유 필드 값을 가져오기
CheckResultMarginFree	요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 margin_free 필드 값을 가져오기
CheckResultMarginLevel	요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 margin_level 필드 값을 가져오기
CheckResultComment	요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 주석 필드 값을 가져오기
마지막 요청 실행 결과에 대한 액세스	
Result	마지막 요청 결과의 구조 복사본을 가져오기
ResultRetcode	요청 결과 코드 가져오기
ResultRetcodeDescription	요청 결과의 코드를 문자열로 가져오기
ResultDeal	딜 티켓 가져오기
ResultOrder	주문 티켓 가져오기
ResultVolume	딜 또는 주문 볼륨 가져오기

© 2000-2025, MetaQuotes Ltd.

매개변수 설정	
ResultPrice	브로커가 확인한 가격 가져오기
ResultBid	현재 입찰가(재요청) 가져오기
ResultAsk	현재 호가(재요청) 가져오기
ResultComment	브로커 설명 가져오기
보조 메서드	
PrintRequest	마지막 요청 매개변수를 저널에 인쇄하기
PrintResult	마지막 요청의 결과를 저널에 인쇄하기
FormatRequest	마지막 요청 매개변수를 사용하여 포맷된 문자열을 준비하기
FormatRequestResult	마지막 요청 실행 결과로 포맷된 문자열을 준비하기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

LogLevel

메시지의 로깅 수준 설정하기.

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // 수준
  \)
```

매개변수

log_level

[in] 새 로깅 수준.

값 반환

없음.

참고

LOG_LEVEL_NO 이하에서는 메시지 표시를 비활성화합니다(최적화 모드에서 자동으로 설정). LOG_LEVEL_ERRORS는 오류 메시지만 표시합니다(기본값). LOG_LEVEL_ALL 이상에서는 모든 메시지를 표시할 수 있습니다(테스트 모드에서 자동으로 설정).

ENUM_LOG_LEVELS

식별자	설명	값
LOG_LEVEL_NO	메시지 표시 사용 안함	0
LOG_LEVEL_ERRORS	오류 메시지만 표시됩니다	1
LOG_LEVEL_ALL	모든 메시지가 표시됩니다	2

© 2000-2025, MetaQuotes Ltd.

SetExpertMagicNumber

엑스퍼트 ID 설정.

```
void  SetExpertMagicNumber(
   ulong  magic      // ID
  \)
```

매개변수

magic

[in] 새 엑스퍼트 ID.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

SetDeviationInPoints

허용된 편차를 설정하기.

```
void  SetDeviationInPoints(
   ulong  deviation      // 편차
  \)
```

매개변수

편차

[in] 허용된 편차(점 단위).

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

SetTypeFilling

주문 채우기 유형 설정하기.

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // 주문 채우기 유형
  \)
```

매개변수

filling

[in] ENUM_ORDER_TYPE_FILLING 열거형에서 주문 채우기 유형.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

SetTypeFillingBySymbol

지정된 기호 설정에 따라 주문의 채우기 유형을 설정하기.

```
bool  SetTypeFillingBySymbol(
   const string   symbol      // 기호 이름
  \)
```

매개변수

Symbol

[in] 기호 이름(SYMBOL_FILLING_MODE에 호용된 주문 채우기 정책이 포함되어 있음).

값 반환

실행에 성공하면 true, 채우기 정책 정의에 실패하면 false.

참고

SYMBOL_FILLING_FOK 및 SYMBOL_FILLING_IOC 채우기 정책이 동시에 허용되는 경우, ORDER_FILLING_FOK 값이 순서에 대해 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

SetAsyncMode

거래 작업에 대한 비동기 모드를 설정하기.

```
void  SetAsyncMode(
   bool  mode      // 비동기 모드 플래그
  \)
```

매개변수

mode

[in] 비동기 모드 플래그.

값 반환

없음.

참고

이 모드는 전송된 요청에 대한 거래 서버의 응답을 기다리지 않고 비동기 거래 작업에 사용됩니다 (OrderSendAsync참조).

© 2000-2025, MetaQuotes Ltd.

SetMarginMode

현재 계정 설정에 따라 마진 계산 모드를 설정하기.

```
void  SetMarginMode()
```

값 반환

No.

참고

마진 계산 모드는 ENUM_ACCOUNT_MARGIN_MODE에 명시되어 있습니다.

© 2000-2025, MetaQuotes Ltd.

OrderOpen

매개변수가 설정된 지정가 주문을 발주하기.

```
bool  OrderOpen(
   const string          symbol,          // 기호
   ENUM_ORDER_TYPE       order_type,      // 주문 유형
   double                volume,          // 주문 볼륨
   double                limit_price,     // StopLimit 가격
   double                price,           // 실행 가격
   double                sl,              // 손절매 가격
   double                tp,              // 이익 실현 가격
   ENUM_ORDER_TYPE_TIME  type_time,       // 만료별 유형
   datetime              expiration,      // 만료
   const string          comment=""       // 코멘트
  \)
```

매개변수

Symbol

[in] 거래 상품 이름.

order_type

[in] ENUM_ORDER_TYPE 열거의 주문 거래 작업 형식.

볼륨

[in] 요청된 주문 볼륨.

limit_price

[in] StopLimit 주문이 발주될 가격.

가격

[in] 주문을 실행해야 하는 가격.

sl

[in] 손절매가 트리거할 가격.

tp

[in] 이익 실현이 트리거할 가격.

type_time

[in] ENUM_ORDER_TYPE_TIME 열거에서 실행별 주문 유형.

expiration

[in] 지정가 주문 만료일.

comment=""

[in] 주문 설명.

값 반환

기본 구조를 성공적으로 확인하면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

참고

OrderSend(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. 다음을 사용하여 거래 요청 결과(거래 서버 반환 코드)를 확인해야 합니다: ResultRetcode() 및 ResultOrder()에서 반환한 값.

© 2000-2025, MetaQuotes Ltd.

OrderModify

지정가 주문 매개변수 수정하기.

```
bool  OrderModify(
   ulong                 ticket,          // 주문 티켓
   double                price,           // 실행 가격
   double                sl,              // 손절매 가격
   double                tp,              // 이익 실현 가격
   ENUM_ORDER_TYPE_TIME  type_time,       // 만료별 유형
   datetime              expiration,      // 만료
   double                stoplimit        // 제한 주문 가격
  \)
```

매개변수

티켓

[in] 주문 티켓.

가격

[in] 주문을 실행해야 하는 새 가격(또는 변경할 필요가 없는 경우 이전 값).

sl

[in] 손절매가 트리거할 새 가격(또는 변경할 필요가 없는 이전 값).

tp

[in] 이익 실현이 트리거할 새 가격(또는 변경할 필요가 없는 경우 이전 값).

type_time

[in] ENUM_ORDER_TYPE_TIME 열거(변경할 필요가 없는 경우 이전 값)의 만료별 새 주문 유형.

expiration

[in] 지정가 주문의 새 만료 날짜(또는 변경할 필요가 없는 경우 이전 값).

stoplimit

[in] 가격이 *가격 값에 도달할 때 제한 주문을 설정할 때 사용되는 새 가격.* 다음에만 지정됩니다: StopLimit 주문.

값 반환

기본 구조를 성공적으로 확인하면 true, 그렇지 않으면 false.

참고

OrderModify(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. ResultRetcode()를 사용하여 거래 결과(거래 서버 반환코드)를 확인해야 합니다.

© 2000-2025, MetaQuotes Ltd.

OrderDelete

지정가 주문 삭제하기.

```
bool  OrderDelete(
   ulong  ticket      // 주문 티켓
  \)
```

매개변수

티켓

[in] 주문 티켓.

값 반환

기본 구조를 성공적으로 확인하면 true, 그렇지 않으면 false.

참고

OrderDelete(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. ResultRetcode()를 사용하여 거래 결과(거래 서버 반환코드)를 확인해야 합니다.

© 2000-2025, MetaQuotes Ltd.

PositionOpen

지정된 매개변수가 있는 포지션을 열기.

```
bool  PositionOpen(
   const string     symbol,          // 기호
   ENUM_ORDER_TYPE  order_type,      // 오픈 포지션을 열기 위한 주문 유형
   double           volume,          // 포지션 볼륨
   double           price,           // 실행 가격
   double           sl,              // 손절매 가격
   double           tp,              // 이익 실현 가격
   const string     comment=""       // 코멘트
  \)
```

매개변수

Symbol

[in] 포지션을 열기 위한 거래 상품 이름.

order_type

[in] ENUM_ORDER_TYPE 열거에서 위치를 여는 주문 유형(거래 작업).

볼륨

[in] 요청된 포지션 볼륨.

가격

[in] 포지션을 열어야 하는 가격.

sl

[in] 손절매가 트리거할 가격.

tp

[in] 이익 실현이 트리거할 가격.

comment=""

[in] 포지션 설명.

값 반환

기본 구조를 성공적으로 확인하면 true, 그렇지 않으면 false.

참고

PositionOpen(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. 다음을 사용하여 거래 요청 결과(거래 서버 반환 코드)를 확인해야 합니다: ResultRetcode() 및 ResultDeal()에서 반환한 값.

© 2000-2025, MetaQuotes Ltd.

PositionModify

지정된 기호별로 포지션 매개변수를 수정하기.

```
bool  PositionModify(
   const string  symbol,      // 기호
   double        sl,          // 손절매 가격
   double        tp           // 이익 실현 가격
  \)
```

지정한 티켓별로 포지션 매개변수를 수정하기.

```
bool  PositionModify(
   const ulong   ticket,      // 포지션 티켓
   double        sl,          // 손절매 가격
   double        tp           // 이익 실현 가격
  \)
```

매개변수

Symbol

[in] 포지션을 수정하기 위한 거래 상품 이름.

티켓

[in] 수정할 포지션 티켓.

sl

[in] 손절매가 트리거할 새 가격(또는 변경할 필요가 없는 이전 값).

tp

[in] 이익 실현이 트리거할 새 가격(또는 변경할 필요가 없는 경우 이전 값).

값 반환

기본 구조를 성공적으로 확인하면 true, 그렇지 않으면 false.

참고

PositionModify(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. ResultRetcode()를 사용하여 거래 결과(거래 서버 반환코드)를 확인해야 합니다.

포지션의 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE) "netting" 해석의 경우, 하나의 포지션만이 Symbol에 대해서 존재할 수 있습니다. 이 포지션은 하나 이상의 딜 결과입니다. 도구 상자 창의 거래 탭에도 표시되는 유효한 지정가 주문의 포지션을 혼동하지 마십시오.

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING),기호 하나에 대해 여러 포지션을 열 수 있습니다. 이 경우 PositionModify가 가장 낮은 티켓으로 포지션을 수정합니다.

© 2000-2025, MetaQuotes Ltd.

PositionClose

지정된 기호로 포지션 닫기.

```
bool  PositionClose(
   const string  symbol,                  // 기호
   ulong         deviation=ULONG_MAX      // 편차
  \)
```

지정된 티켓으로 포지션 닫기.

```
bool  PositionClose(
   const ulong   ticket,                  // 포지션 티켓
   ulong         deviation=ULONG_MAX      // 편차
  \)
```

매개변수

Symbol

[in] 포지션을 닫을 거래 상품 이름.

티켓

[in] 닫힌 포지션 티켓.

deviation=ULONG_MAX

[in] 현재 가격과의 최대 편차(점 단위).

값 반환

기본 구조를 성공적으로 확인하면 true, 그렇지 않으면 false.

참고

PositionClose(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. ResultRetcode()를 사용하여 거래 결과(거래 서버 반환코드)를 확인해야 합니다.

포지션의 (ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE) "netting" 해석의 경우, 하나의 포지션만이 Symbol에 대해서 존재할 수 있습니다. 이 포지션은 하나 이상의 딜 결과입니다. 도구 상자 창의 거래 탭에도 표시되는 유효한 지정가 주문의 포지션을 혼동하지 마십시오.

개별 포지션이 허용되는 경우 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING),기호 하나에 대해 여러 포지션을 열 수 있습니다. 이 경우, PositionClose가 가장 낮은 티켓으로 포지션을 닫습니다.

© 2000-2025, MetaQuotes Ltd.

PositionClosePartial

"hedging" 어카운팅의 경우 지정된 기호에서 포지션을 부분적으로 닫기.

```
bool  PositionClosePartial(
   const string  symbol,                  // 기호
   const double  volume,                  // 볼륨
   ulong         deviation=ULONG_MAX      // 편차
  \)
```

"hedging" 어카운팅의 경우 지정된 티켓이 있는 포지션을 부분적으로 닫기.

```
bool  PositionClosePartial(
   const ulong   ticket,                  // 포지션 티켓
   const double  volume,                  // 볼륨
   ulong         deviation=ULONG_MAX      // 편차
  \)
```

매개변수

Symbol

[in] 포지션이 부분적으로 마감된 거래 상품의 이름. 포지션 부분 닫기에 대해 기호(티켓 아님)가 지정된 경우 기호에 지정된 MagicNumber (엑스퍼트 어드바이저 ID)가 있는 첫 번째 감지 포지션이 지정됩니다. 따라서 지정된 위치 티켓과 함께 PositionClosePartial()을 사용하는 것이 더 나을 수 있습니다.

볼륨

[in] 포지션을 줄여야 하는 볼륨. 값이 부분적으로 닫힌 포지션의 볼륨을 초과하면 완전히 닫힙니다. 반대 포지션이 열리지 않음

티켓

[in] 닫힌 포지션 티켓.

deviation=ULONG_MAX

[in] 현재 가격과의 최대 편차(점 단위).

값 반환

구조의 기본 검사가 성공적이면 true, 그렇지 않으면 false.

참고

PositionClosePartial(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. ResultRetcode() 메서드를 호출하여 거래 요청(거래 서버 반환 코드) 결과를 확인해야 합니다.

"netting" 시스템(ACCOUNT_MARGIN_MODE_RETAIL_NETTING 및 ACCOUNT_MARGIN_MODE_EXCHANGE)에서는 각 Symbol에 대해 주어진 시점에 하나의 포지션(하나 이상의 딜의 결과)만 열수 있습니다. 현재 지정가 주문을 클라이언트 터미널의 "Toolbox" 패널의 "Trade" 탭에도 표시되는 포지션과 혼동하지 마십시오.

포지션 표시 (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)의 경우, 각 기호에서 여러 포지션을 동시에 열 수 있습니다. 이 경우 PositionClose는 최소 티켓이 있는 모지션을 닫습니다.

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

PositionCloseBy

지정된 티켓과 반대 포지션으로 포지션을 닫기.

```
bool  PositionCloseBy(
   const ulong   ticket,         // 포지션 티켓
   const ulong   ticket_by       // 반대 포지션 티켓
  \)
```

매개변수

티켓

[in] 닫힌 포지션의 티켓.

ticket_by

[in] 마감에 사용되는 반대 포지션의 티켓.

반환된 값

구조의 기본 검사가 성공적이면 true, 그렇지 않으면 false.

참고

PositionCloseBy(...) 메서드가 성공적으로 완료되었다고 해서 반드시 거래 작업이 성공적으로 실행되는 것은 아닙니다. ResultRetcode() 메서드를 호출하여 거래 요청(거래 서버 반환 코드) 결과를 확인해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Buy

지정된 매개변수가 있는 long 포지션 열기.

```
bool  Buy(
   double         volume,           // 포지션 볼륨
   const string   symbol=NULL,      // 기호
   double         price=0.0,        // 실행 가격
   double         sl=0.0,           // 손절매 가격
   double         tp=0.0,           // 이익 실현 가격
   const string   comment=""        // 코멘트
  \)
```

매개변수

볼륨

[in] 요청된 포지션 볼륨.

symbol=NULL

[in] 포지션 기호. 지정하지 않으면 현재 기호가 사용됩니다.

price=0.0

[in] 가격. 가격이 지정되지 않은 경우 현재 시장 호가가 사용됩니다.

sl=0.0

[in] 손절매 가격.

tp=0.0

[in] 이익 실현 가격.

comment=""

[in] 코멘트.

값 반환

구조 확인이 성공적이면 true, 그렇지 않으면 false.

참고

Buy(...) 메서드가 성공적으로 완료되었다고 해서 반드시 거래 작업이 성공적으로 실행되는 것은 아닙니다. 다음을 사용하여 거래 요청 결과(거래 서버 반환 코드)를 확인해야 합니다: ResultRetcode() 및 ResultDeal()에서 반환한 값.

© 2000-2025, MetaQuotes Ltd.

Sell

지정된 매개변수로 short 포지션 열기.

```
bool  Sell(
   double         volume,           // 포지션 볼륨
   const string   symbol=NULL,      // 기호
   double         price=0.0,        // 실행 가격
   double         sl=0.0,           // 손절매 가격
   double         tp=0.0,           // 이익 실현 가격
   const string   comment=""        // 코멘트
  \)
```

매개변수

볼륨

[in] 요청된 포지션 볼륨.

symbol=NULL

[in] 포지션 기호. 기호를 지정하지 않으면 현재 기호가 사용됩니다.

price=0.0

[in] 가격. 가격이 지정되지 않은 경우 현재 시장 입찰가가 사용됩니다.

sl=0.0

[in] 손절매 가격.

tp=0.0

[in] 이익 실현 가격.

comment=""

[in] 코멘트.

값 반환

구조 확인이 성공적이면 true, 그렇지 않으면 false.

참고

Sell(...) 메서드가 성공적으로 완료되었다고 해서 반드시 거래 작업이 성공적으로 실행되는 것은 아닙니다. 다음을 사용하여 거래 요청 결과(거래 서버 반환 코드)를 확인해야 합니다: ResultRetcode() 및 ResultDeal()에서 반환한 값.

© 2000-2025, MetaQuotes Ltd.

BuyLimit

지정된 매개변수를 사용하여 매수 한도 유형(현재 시장 가격보다 낮은 가격으로 매수)의 지정가 주문 발주하기.

```
bool  BuyLimit(
   double                volume,                    // 주문 볼륨
   double                price,                     // 주문 가격
   const string          symbol=NULL,               // 기호
   double                sl=0.0,                    // 손절매 가격
   double                tp=0.0,                    // 이익 실현 가격
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // 주문 일생
   datetime              expiration=0,              // 주문 만료 시간
   const string          comment=""                 // 코멘트
  \)
```

매개변수

볼륨

[in] 요청된 주문 볼륨.

가격

[in] 주문 가격.

symbol=NULL

[in] 주문 가격. 기호를 지정하지 않으면 현재 기호가 사용됩니다.

sl=0.0

[in] 손절매 가격.

tp=0.0

[in] 이익 실현 가격.

type_time=ORDER_TIME_GTC

[in] ENUM_ORDER_TYPE_TIME 열거에서 주문 일생.

expiration=0

[in] 주문 만료 시간(type_time=ORDER_TIME_SPECIFIED 인 경우에만 사용됨).

comment=""

[in] 주문 설명.

값 반환

구조 확인이 성공적이면 true, 그렇지 않으면 false.

참고

BuyLimit(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. 다음을 사용하여 거래 요청 결과(트레이드 서버 반환 코드)를 확인해야 합니다 :ResultRetcode() 및 ResultOrder()에서 반환한 값.

© 2000-2025, MetaQuotes Ltd.

BuyStop

지정된 매개변수를 사용하여 매수 중지 유형(현재 시장 가격보다 높은 가격으로 구입)의 지정가 주문 발주하기.

```
bool  BuyStop(
   double                volume,                    // 주문 볼륨
   double                price,                     // 주문 가격
   const string          symbol=NULL,               // 기호
   double                sl=0.0,                    // 손절매 가격
   double                tp=0.0,                    // 이익 실현 가격
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // 주문 일생
   datetime              expiration=0,              // 주문 만료 시간
   const string          comment=""                 // 코멘트
  \)
```

매개변수

볼륨

[in] 요청된 주문 볼륨.

가격

[in] 주문 가격.

symbol=NULL

[in] 주문 가격. 기호를 지정하지 않으면 현재 기호가 사용됩니다.

sl=0.0

[in] 손절매 가격.

tp=0.0

[in] 이익 실현 가격.

type_time=ORDER_TIME_GTC

[in] ENUM_ORDER_TYPE_TIME 열거에서 주문 일생.

expiration=0

[in] 주문 만료 시간(type_time=ORDER_TIME_SPECIFIED 인 경우에만 사용됨).

comment=""

[in] 주문 설명.

값 반환

구조 확인이 성공적이면 true, 그렇지 않으면 false.

참고

BuyStop(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. 다음을 사용하여 거래 요청 결과(트레이드 서버 반환 코드)를 확인해야 합니다 :ResultRetcode() 및 ResultOrder()에서 반환한 값.

© 2000-2025, MetaQuotes Ltd.

SellLimit

지정된 매개변수를 사용하여 판매 제한 유형(현재 시장 가격보다 높은 가격으로 판매)의 지정가 주문을 발주하기.

```
bool  SellLimit(
   double                volume,                    // 주문 볼륨
   double                price,                     // 주문 가격
   const string          symbol=NULL,               // 기호
   double                sl=0.0,                    // 손절매 가격
   double                tp=0.0,                    // 이익 실현 가격
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // 주문 일생
   datetime              expiration=0,              // 주문 만료 시간
   const string          comment=""                 // 코멘트
  \)
```

매개변수

볼륨

[in] 요청된 주문 볼륨.

가격

[in] 주문 가격.

symbol=NULL

[in] 주문 가격. 기호를 지정하지 않으면 현재 기호가 사용됩니다.

sl=0.0

[in] 손절매 가격.

tp=0.0

[in] 이익 실현 가격.

type_time=ORDER_TIME_GTC

[in] ENUM_ORDER_TYPE_TIME 열거에서 주문 일생.

expiration=0

[in] 주문 만료 시간(type_time=ORDER_TIME_SPECIFIED 인 경우에만 사용됨).

comment=""

[in] 주문 설명.

값 반환

구조 확인이 성공적이면 true, 그렇지 않으면 false.

참고

SellLimit(...) 메서드가 성공적으로 완료되었다고 해서 항상 트레이드 작업이 성공적으로 실행되는 것은 아닙니다. 다음을 사용하여 거래 요청 결과(트레이드 서버 반환 코드)를 확인해야 합니다 :ResultRetcode() 및 ResultOrder()에서 반환한 값.

© 2000-2025, MetaQuotes Ltd.

SellStop

지정된 매개변수를 사용하여 판매 중지 유형(현재 시장 가격보다 낮은 가격으로 판매)의 지정가 주문을 발주합니다.

```
bool  SellStop(
   double                volume,                     // 주문 볼륨
   double                price,                      // 주문 가격
   const string          symbol=NULL,                // 기호
   double                sl=0.0,                     // 손절매 가격
   double                tp=0.0,                     // 이익 실현 가격
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,   // 주문 일생
   datetime              expiration=0,               // 주문 만료 시간
   const string          comment=""                  // 코멘트
  \)
```

매개변수

볼륨

[in] 요청된 주문 볼륨.

가격

[in] 주문 가격.

symbol=NULL

[in] 주문 가격. 기호를 지정하지 않으면 현재 기호가 사용됩니다.

sl=0.0

[in] 손절매 가격.

tp=0.0

[in] 이익 실현 가격.

type_time=ORDER_TIME_GTC

[in] ENUM_ORDER_TYPE_TIME 열거에서 주문 일생.

expiration=0

[in] 주문 만료 시간(type_time=ORDER_TIME_SPECIFIED 인 경우에만 사용됨).

comment=""

[in] 주문 설명.

값 반환

구조 확인이 성공적이면 true, 그렇지 않으면 false.

참고

SellStop(...) 메서드가 성공적으로 완료되었다고 해서 항상 거래 작업이 성공적으로 실행되는 것은 아닙니다. 다음을 사용하여 거래 요청 결과(트레이드 서버 반환 코드)를 확인해야 합니다 :ResultRetcode() 및 ResultOrder()에서 반환한 값.

© 2000-2025, MetaQuotes Ltd.

Request

마지막 요청 구조의 복사본 가져오기.

```
void  Request(
   MqlTradeRequest&  request      // 참조
   ) const
```

매개변수

요청

[out] 복사된 구조에 대한 참조.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

RequestAction

거래 작업 유형 가져오기.

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

값 반환

거래 작업 유형: ENUM_TRADE_REQUEST_ACTIONS 열거의 마지막 요청에 사용된 거래 작업 유형.

© 2000-2025, MetaQuotes Ltd.

RequestActionDescription

거래 작업 유형을 문자열로 가져오기.

```
string  RequestActionDescription() const
```

값 반환

거래 작업 유형(문자열): 마지막 작업에서 사용됨.

© 2000-2025, MetaQuotes Ltd.

RequestMagic

엑스퍼트 어드바이저의 매직 넘버를 가져오기.

```
ulong  RequestMagic() const
```

값 반환

엑스퍼트 어드바이저의 매직 넘버 (ID): 마지막 요청에 사용됨.

© 2000-2025, MetaQuotes Ltd.

RequestOrder

주문 티켓 가져오기.

```
ulong  RequestOrder() const
```

값 반환

마지막 요청의 주문 티켓.

© 2000-2025, MetaQuotes Ltd.

RequestSymbol

기호의 이름 가져오기.

```
string  RequestSymbol() const
```

값 반환

마지막 요청에 사용된 기호 이름.

© 2000-2025, MetaQuotes Ltd.

RequestVolume

거래 볼륨(로트 단위) 가져오기.

```
double  RequestVolume() const
```

값 반환

마지막 요청에 사용된 요청된 거래 볼륨(로트 단위).

© 2000-2025, MetaQuotes Ltd.

RequestPrice

주문 실행 가격 가져오기.

```
double  RequestPrice() const
```

값 반환

마지막 요청에 사용된 주문 실행 가격.

© 2000-2025, MetaQuotes Ltd.

RequestStopLimit

중지 제한 유형의 지정가 주문 가격 가져오기.

```
double  RequestStoplimit() const
```

값 반환

마지막 요청에 사용된 중지 제한 유형의 지정가 주문 가격.

© 2000-2025, MetaQuotes Ltd.

RequestSL

주문의 손절매 가격 가져오기.

```
double  RequestSL() const
```

값 반환

마지막 요청에 사용된 손절매 가격.

© 2000-2025, MetaQuotes Ltd.

RequestTP

주문 이익 실현 가격을 가져오기.

```
double  RequestTP() const
```

값 반환

이익 실현 가격: 마지막 요청에 사용됨.

© 2000-2025, MetaQuotes Ltd.

RequestDeviation

요청한 가격에서 최대 가격 편차를 가져오기.

```
ulong  RequestDeviation() const
```

값 반환

최대 허용 가격 편차: 마지막 요청에서 사용됨.

© 2000-2025, MetaQuotes Ltd.

RequestType

주문의 유형 가져오기.

```
ENUM_ORDER_TYPE  RequestType() const
```

값 반환

ENUM_ORDER_TYPE 열거의 마지막 요청에 사용된 주문 유형.

© 2000-2025, MetaQuotes Ltd.

RequestTypeDescription

주문 유형(문자열) 가져오기.

```
string  RequestTypeDescription() const
```

값 반환

마지막 요청에 사용된 주문 유형(문자열).

© 2000-2025, MetaQuotes Ltd.

RequestTypeFilling

주문 채우기 유형 가져오기.

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

값 반환

채우기 유형 : 마지막 요청에 사용된 ENUM_ORDER_TYPE_FILLING의 주문.

© 2000-2025, MetaQuotes Ltd.

RequestTypeFillingDescription

문자열로 주문 채우기 유형 가져오기.

```
string  RequestTypeFillingDescription() const
```

값 반환

채우기 유형(문자열): 마지막 요청에 사용된 주문.

© 2000-2025, MetaQuotes Ltd.

RequestTypeTime

주문의 유효 기간 가져오기.

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

값 반환

마지막 요청에 사용된 ENUM_ORDER_TYPE_TIME 열거형의 주문 유효 기간.

© 2000-2025, MetaQuotes Ltd.

RequestTypeTimeDescription

주문의 유효 기간(문자열로) 가져오기.

```
string  RequestTypeTimeDescription() const
```

값 반환

주문의 유효기간(문자열): 마지막 요청에 사용됨.

© 2000-2025, MetaQuotes Ltd.

RequestExpiration

지정가 주문의 만료 시간 가져오기.

```
datetime  RequestExpiration() const
```

값 반환

마지막 요청에 사용된 지정가 주문의 만료 시간.

© 2000-2025, MetaQuotes Ltd.

RequestComment

주문의 설명 가져오기.

```
string  RequestComment() const
```

값 반환

마지막 요청에 사용된 주문의 설명.

© 2000-2025, MetaQuotes Ltd.

RequestPosition

포지션 티켓 가져오기.

```
ulong  RequestPosition() const
```

값 반환

마지막 요청에 사용된 포지션 티켓.

© 2000-2025, MetaQuotes Ltd.

RequestPositionBy

반대 포지션 티켓 가져오기.

```
ulong  RequestPositionBy() const
```

값 반환

마지막 요청에 사용된 반대 포지션 티켓.

© 2000-2025, MetaQuotes Ltd.

Result

마지막 요청 결과의 구조 복사본 가져오기.

```
void  Result(
   MqlTradeResult&  result      // 참조
   ) const
```

매개변수

결과

[out] 복사를 위한 MqlTradeResult 유형 구조 참조.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

ResultRetcode

요청 실행 결과의 코드 가져오기.

```
uint  ResultRetcode() const
```

값 반환

요청 결과의 코드.

© 2000-2025, MetaQuotes Ltd.

ResultRetcodeDescription

요청 실행 결과의 코드를 문자열로 가져오기.

```
string  ResultRetcodeDescription() const
```

값 반환

***마지막 요청 결과의 코드 문자열로서의.

© 2000-2025, MetaQuotes Ltd.

ResultDeal

딜 티켓 가져오기.

```
ulong  ResultDeal() const
```

값 반환

딜이 실행된 경우 딜 티켓.

© 2000-2025, MetaQuotes Ltd.

ResultOrder

주문 티켓 가져오기.

```
ulong  ResultOrder() const
```

값 반환

주문이 발주된 경우 주문 티켓.

© 2000-2025, MetaQuotes Ltd.

ResultVolume

딜 또는 주문 볼륨 가져오기.

```
double  ResultVolume() const
```

값 반환

브로커에 의해 확인된 딜 또는 주문 볼륨.

© 2000-2025, MetaQuotes Ltd.

ResultPrice

브로커가 확인한 가격 가져오기.

```
double  ResultPrice() const
```

값 반환

브로커가 확인한 가격.

© 2000-2025, MetaQuotes Ltd.

ResultBid

현재 입찰가(재요청) 가져오기.

```
double  ResultBid() const
```

값 반환

현재 입찰가(재요청).

© 2000-2025, MetaQuotes Ltd.

ResultAsk

현재 호가(재요청) 가져오기.

```
double  ResultAsk() const
```

값 반환

현재 호가(재요청).

© 2000-2025, MetaQuotes Ltd.

ResultComment

브로커 설명 가져오기.

```
string  ResultComment() const
```

값 반환

작업에 대한 브로커 설명.

© 2000-2025, MetaQuotes Ltd.

CheckResult

마지막 요청 검사 결과의 구조 복사본 가져오기.

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // 참조
   ) const
```

매개변수

check_result

[out] 복사할 MqlTradeCheckResult 유형의 대상 구조에 대한 참조.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

CheckResultRetcode

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 retcode 필드 값을 가져오기.

```
uint  CheckResultRetcode() const
```

값 반환

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 retcode 필드 값(오류 코드)입니다.

© 2000-2025, MetaQuotes Ltd.

CheckResultRetcodeDescription

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 retcode 필드에 대한 문자열 설명 가져오기.

```
string  ResultRetcodeDescription() const
```

값 반환

요청 정확성을 확인하는 동안 MqlTradeCheckResult 유형의 retcode 필드(오류 코드)에 대한 문자열 설명이 채워졌습니다.

© 2000-2025, MetaQuotes Ltd.

CheckResultBalance

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형 밸런스 필드 값을 가져오기.

```
double  CheckResultBalance() const
```

값 반환

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 균형 필드 값(거래 작업 실행 후 발생할 밸런스 값)입니다.

© 2000-2025, MetaQuotes Ltd.

CheckResultEquity

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 자기자본 필드 값을 가져오기.

```
double  CheckResultEquity() const
```

값 반환

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 자기자본 필드 값(거래 작업 실행 후 발생할 지분 값)입니다.

© 2000-2025, MetaQuotes Ltd.

CheckResultProfit

플로팅(floating) 이익 값 가져오기.

```
double  CheckResultProfit() const
```

값 반환

거래 작업을 수행한 후 플로팅(floating) 이익 값.

© 2000-2025, MetaQuotes Ltd.

CheckResultMargin

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 마진 필드 값을 가져오기.

```
double  CheckResultMargin() const
```

값 반환

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 마진 필드 값(거래 작업에 필요한 마진)입니다.

© 2000-2025, MetaQuotes Ltd.

CheckResultMarginFree

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 margin_free 필드 값을 가져오기

```
double  CheckResultMarginFree() const
```

값 반환

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 margin_free 필드 값(거래 작업 실행 후 남는 여유 마진)입니다.

© 2000-2025, MetaQuotes Ltd.

CheckResultMarginLevel

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 margin_level 필드 값을 가져오기.

```
double  CheckResultMarginLevel() const
```

값 반환

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 margin_level 필드 값(거래 작업 실행 후 설정될 마진 레벨)입니다.

© 2000-2025, MetaQuotes Ltd.

CheckResultComment

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 주석 필드 값.

```
string  CheckResultComment() const
```

값 반환

요청 정확성을 확인하는 동안 채워진 MqlTradeCheckResult 유형의 주석 필드 값(응답 코드에 대한 설명, 오류 설명)입니다.

© 2000-2025, MetaQuotes Ltd.

PrintRequest

마지막 요청 매개변수를 저널에 인쇄하기.

```
void  PrintRequest() const
```

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

PrintResult

마지막 요청의 결과를 저널에 인쇄하기.

```
void  PrintResult() const
```

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

FormatRequest

마지막 요청 매개변수를 사용하여 포맷된 문자열을 준비하기.

```
string  FormatRequest(
   string&                  str,          // 문자열
   const MqlTradeRequest&   request      // 요청
   ) const
```

매개변수

str

[in] 참조에서 전달된 대상 문자열.

요청

[in] 마지막 요청의 매개변수가 있는 MqlTradeRequest 유형 구조.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

FormatRequestResult

마지막 요청 실행 결과로 포맷된 문자열을 준비하기.

```
string  FormatRequestResult(
   string&                 str,          // 문자열
   const MqlTradeRequest&  request,      // 요청
   const MqlTradeResult&   result        // 결과
   ) const
```

매개변수

str

[in] 참조에서 전달된 대상 문자열.

요청

[in] 마지막 요청의 매개변수가 있는 MqlTradeRequest 유형의 구조.

결과

[in] 마지막 요청 결과가 포함된 MqlTradeResult 유형의 구조.

값 반환

없음.

© 2000-2025, MetaQuotes Ltd.

CTerminalInfo

CTerminalInfo는 mql5 프로그램 환경의 속성에 대한 단순 액세스를 위한 클래스입니다.

설명

CTerminalInfo 클래스는 mql5 프로그램 환경 속성에 대한 액세스를 제공합니다.

선언

```
class CTerminalInfo : public CObject
```

제목

```
#include <Trade\TerminalInfo.mqh>
```

상속 계층

CObject

CTerminalInfo

그룹별 클래스 메서드

정수 유형의 속성에 액세스하는 메서드	
Build	클라이언트 터미널의 빌드 번호 가져오기
IsConnected	거래 서버 연결에 대한 정보 가져오기
IsDLLsAllowed	DLL 사용 권한에 대한 정보 가져오기
IsTradeAllowed	거래 권한에 대한 정보 가져오기
IsEmailEnabled	터미널에 지정된 SMTP 서버에 이메일을 보내고 로그인할 수 있는 권한에 대한 정보 가져오기
IsFtpEnabled	터미널 설정에 지정된 FTP 서버에 거래 보고서를 보내고 로그인할 수 있는 권한에 대한 정보를 가져오기
MaxBars	차트의 최대 막대 수에 대한 정보 가져오기
CodePage	클라이언트 터미널에서 언어의 코드 페이지에 대한 정보 가져오기
CPUCores	CPU 코어에 대한 정보 가져오기
MemoryPhysical	실제 메모리에 대한 정보 가져오기(MB 단위)
MemoryTotal	터미널/에이전트 프로세스에 사용할 수 있는 총 메모리에 대한 정보 가져오기(MB 단위)
MemoryAvailable	터미널/에이전트 프로세스에 사용할 수 있는 사용 가능 메모리에 대한 정보 가져오기(MB 단위)
MemoryUsed	터미널/에이전트 프로세스에서 사용하는 메모리에 대한 정보 가져오기(MB 단위)

© 2000-2025, MetaQuotes Ltd.

정수 유형의 속성에 액세스하는 메서드	
IsX64	클라이언트 터미널 유형에 대한 정보 가져오기
OpenCLSupport	비디오 카드에서 지원되는 OpenCL 버전에 대한 정보 가져오기
DiskSpace	사용 가능한 디스크 공간에 대한 정보 가져오기(MB 단위)
문자열 유형의 속성에 액세스하는 방법	
Language	클라이언트 터미널의 언어 가져오기
Name	클라이언트 터미널 이름 가져오기
Company	클라이언트 터미널의 회사 이름 가져오기
Path	클라이언트 터미널의 폴더 가져오기
DataPath	클라이언트 터미널의 데이터 폴더 가져오기
CommonDataPath	컴퓨터에 설치된 모든 클라이언트 터미널의 공통 데이터 폴더를 가져오기
MQL5 API 기능에 액세스	
InfoInteger	정수 유형의 속성 값
InfoString	문자열 형식의 속성 값 가져오기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Build

클라이언트 터미널의 빌드 번호 가져오기.

```
int  CBuild() const
```

값 반환

클라이언트 터미널의 빌드 번호.

참고

터미널 빌드 번호는 TerminalInfoInteger() 함수에 의해 정의 됩니다(TERMINAL_BUILD 속성).

© 2000-2025, MetaQuotes Ltd.

IsConnected

거래 서버 연결에 대한 정보 가져오기.

```
bool  IsConnected() const
```

값 반환

거래 서버에 터미널이 연결되면 true, 그렇지 않으면 false.

참고

연결 상태는 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_CONNECTED 속성).

© 2000-2025, MetaQuotes Ltd.

IsDLLsAllowed

DLL 사용 권한에 대한 정보 가져오기.

```
bool  IsDLLsAllowed() const
```

값 반환

DLL 사용이 허용되면 true, 그렇지 않으면 false.

참고

DLL 사용 권한은 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_DLLS_ALLOWED 속성).

© 2000-2025, MetaQuotes Ltd.

IsTradeAllowed

거래 권한에 대한 정보 가져오기.

```
bool  IsTradeAllowed() const
```

값 반환

거래가 허용되면 true, 그렇지 않으면 false.

참고

거래 권한은 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_TRADE_ALLOWED 속성).

© 2000-2025, MetaQuotes Ltd.

IsEmailEnabled

터미널 설정에 지정된 SMTP 서버로 이메일을 보내고 로그인할 수 있는 권한에 대한 정보 가져오기.

```
bool  IsEmailEnabled() const
```

값 반환

이메일 전송이 허용되면 true, 그렇지 않으면 false.

참고

이메일 전송 권한은 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_EMAIL_ENABLED 속성).

© 2000-2025, MetaQuotes Ltd.

IsFtpEnabled

터미널 설정에 지정된 FTP 서버에 거래 보고서를 보내고 로그인할 수 있는 권한에 대한 정보를 가져오기.

```
bool  IsFtpEnabled() const
```

값 반환

FTP 서버에 거래 보고서를 전송이 허용되면 true, 그렇지 않으면 false.

참고

거래 보고서 전송 권한은 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_FTP_ENABLED 속성).

© 2000-2025, MetaQuotes Ltd.

MaxBars

터미널 설정에 지정된 차트의 최대 막대 수를 가져오기.

```
int  MaxBars() const
```

값 반환

차트의 최대 막대 수.

참고

차트의 최대 막대 수는 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_MAXBARS 속성).

© 2000-2025, MetaQuotes Ltd.

CodePage

터미널에 있는 언어의 코드 페이지에 대한 정보 가져오기.

```
int  CodePage() const
```

값 반환

클리아언트 터미널에서 언어의 코드 페이지.

참고

언어의 코드 페이지는 TerminalInfoInteger() 함수로 정의됩니다(TERMINAL_CODEPAGE 속성).

© 2000-2025, MetaQuotes Ltd.

CPUCores

시스템의 CPU 코어 수에 대한 정보 가져오기.

```
int  CPUCores() const
```

값 반환

시스템의 CPU 코어 수.

참고

CPU 코어 수는 TerminalInfoInteger() 함수에 의해 정의됩니다 (TERMINAL_CPU_CORES 속성).

© 2000-2025, MetaQuotes Ltd.

MemoryPhysical

실제 메모리에 대한 정보 가져오기(MB 단위).

```
int  MemoryPhysical() const
```

값 반환

실제 메모리(MB 단위).

참고

실제 메모리는 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_MEMORY_PHYSICAL 속성).

© 2000-2025, MetaQuotes Ltd.

MemoryTotal

터미널/에이전트에서 사용할 수 있는 총 메모리 정보를 가져오기(MB 단위).

```
int  MemoryTotal() const
```

값 반환

터미널/에이전트에서 사용할 수 있는 총 메모리(MB 단위).

참고

터미널/에이전트에서 사용할 수 있는 총 메모리는 TerminalInfoInteger() 함수에 의해 정의됩니다 (TERMINAL_MEMORY_TOTAL 속성).

© 2000-2025, MetaQuotes Ltd.

MemoryAvailable

클라이언트 터미널/에이전트에서 사용할 수 있는 사용 가능한 메모리에 대한 정보를 가져오기(MB 단위).

```
int  MemoryTotal() const
```

값 반환

터미널/에이전트에서 사용 가능한 메모리(MB 단위).

참고

클라이언트 터미널/에이전트에서 사용 가능한 메모리는 TerminalInfoInteger() 함수에 의해 정의됩니다 (TERMINAL_MEMORY_TOTAL 속성).

© 2000-2025, MetaQuotes Ltd.

MemoryUsed

클라이언트 터미널/에이전트에서 사용하는 메모리에 대한 정보 가져오기(MB 단위).

```
int  MemoryUsed() const
```

값 반환

클라이언트 터미널/에이전트에서 사용하는 메모리(MB 단위).

참고

클라이언트 터미널/에이전트에서 사용하는 메모리는 TerminalInfoInteger() 함수에 의해 정의됩니다 (TERMINAL_MEMORY_USED 속성).

© 2000-2025, MetaQuotes Ltd.

IsX64

클라이언트 터미널 유형에 대한 정보 가져오기.

```
bool  IsX64() const
```

값 반환

64비트 버전이 사용되면 true, 그렇지 않으면 false.

참고

터미널 유형은 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_X64 속성).

© 2000-2025, MetaQuotes Ltd.

OpenCLSupport

비디오 카드에서 지원되는 OpenCL 버전에 대한 정보 가져오기.

```
int  OpenCLSupport() const
```

값 반환

다음 양식을 가진 OpenCL 버전: 0x00010002 = "1.2". 0은 OpenCL이 지원되지 않음을 의미합니다.

참고

OpenCL 버전은 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_OPENCL_SUPPORT 속성).

© 2000-2025, MetaQuotes Ltd.

DiskSpace

터미널/에이전트의 MQL5₩Files 폴더에 사용할 수 있는 사용 가능 디스크 공간에 대한 정보 가져오기(MB 단위).

```
int  MDiskSpace() const
```

값 반환

사용 가능 디스크 공간.

참고

사용 가능 디스크 공간은 TerminalInfoInteger() 함수에 의해 정의됩니다(TERMINAL_DISK_SPACE 속성).

© 2000-2025, MetaQuotes Ltd.

Language

클라이언트 터미널의 언어에 대한 정보 가져오기.

```
string  Language() const
```

값 반환

클라이언트 터미널에서 사용되는 언어.

참고

터미널 언어는 TerminalInfoString() 함수에 의해 정의됩니다(TERMINAL_LANGUAGE 속성).

© 2000-2025, MetaQuotes Ltd.

Name

클라이언트 터미널 이름에 대한 정보 가져오기.

```
string  Name() const
```

값 반환

클라이언트 터미널 이름.

참고

터미널 이름은 TerminalInfoString() 함수에 의해 정의됩니다(TERMINAL_NAME 속성).

© 2000-2025, MetaQuotes Ltd.

Company

회사 이름에 대한 정보 가져오기.

```
string  Company() const
```

값 반환

회사 이름.

참고

회사 이름은 TerminalInfoString() 함수로 정의됩니다(TERMINAL_COMPANY 속성).

© 2000-2025, MetaQuotes Ltd.

Path

클라이언트 터미널 폴더 가져오기.

```
string  Path() const
```

값 반환

클라이언트 터미널 폴더.

참고

클라이언트 터미널 폴더는 TerminalInfoString() 함수에 의해 사용됩니다 (TERMINAL_PATH 속성).

© 2000-2025, MetaQuotes Ltd.

DataPath

터미널 데티어 폴더에 대한 정보 가져오기.

```
string  DataPath() const
```

값 반환

클라이언트 터미널 데이터 폴더.

참고

클라이언트 터미널은 TerminalInfoString() 삼수에 의해 정의됩니다(TERMINAL_DATA_PATH 속성).

© 2000-2025, MetaQuotes Ltd.

CommonDataPath

컴퓨터에 설치된 모든 클라이언트 터미널의 공통 데이터 폴더를 가져오기.

```
string  CommonDataPath() const
```

값 반환

설치된 모든 터미널의 공통 데이터 폴더.

참고

설치된 모든 터미널의 공통 데이터 폴더는 TerminalInfoString() 함수에 의해 정의됩니다 (COMMON_DATA_PATH 속성).

© 2000-2025, MetaQuotes Ltd.

InfoInteger

mql5 프로그램 환경의 해당 정수 속성 값을 반환.

```
int  TerminalInfoInteger(
   int  property_id      // 속성 식별자
   );
```

매개변수

property_id

[in] 속성 식별자. ENUM_TERMINAL_INFO_INTEGER열거의 값 중 하나일 수 있습니다.

값 반환

int 유형 값.

참고

속성 값은 TerminalInfoInteger() 함수에 의해 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

InfoString

함수는 mql5 프로그램 환경의 해당 속성 값을 반환. 속성은 문자열 유형이어야 합니다.

```
string  TerminalInfoString(
   int  property_id      // 속성 ID
   );
```

매개변수

property_id

[in] 속성 식별자. ENUM_TERMINAL_INFO_STRING 열거 값 중 하나일 수 있습니다.

값 반환

문자열 유형 값.

참고

속성 값은 TerminalInfoString() 함수에 의해 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

거래전략 수립 및 시험수업

이 절에는 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명과 거래 전략의 작성 및 테스트를 위한 클래스 작업에 대한 기술적 세부 정보가 포함되어 있습니다.

이러한 클래스를 사용하면 거래 전략을 만들 때(특히 테스트) 시간을 절약할 수 있습니다.

MQL5 표준 라이브러리(거래 전략 측면)는 터미널 디렉터리인 Include₩Expert 에 배치됩니다.

기본 클래스	Description
CExpertBase	모든 거래 전략 클래스의 기본 클래스
CExpert	Expert Advisor용 기본 클래스
CExpertSignal	거래 시그널 클래스의 기본 클래스
CExpertTrailing	추적 손절매 클래스의 기본 클래스
CExpertMoney	자금 관리 클래스의 기본 클래스

Trading signal classes	Description
CSignalAC	지표 가속 발진기 시장 모델에 기반한 시그널 모듈.
CSignalAMA	적응 이동 평균 지표의 시장 모델에 기초한 시그널 모듈.
CSignalAO	Awesome Oxilator 지표의 시장 모델에 기초한 시그널 모듈.
CSignalBearsPower	발진기 Bears Power의 시장 모델에 기반한 신호 모듈.
CSignalBullsPower	오실레이터 Bulls Power의 시장 모델에 기반한 신호 모듈.
CSignalCCI	발진기 상품채널지수 시장모델에 기초한 시그널 모듈.
CSignalDeM	오실레이터 DeMarker의 시장 모델에 기반한 시그널 모듈.
CSignalDEMA	이중 지수 이동 평균 지표의 시장 모델에 기초한 시그널 모듈.
CSignalEnvelopes	Envelope 지표의 시장 모델에 기초한 시그널 모듈.
CSignalFrAMA	프랙탈 적응형 이동 평균 지표의 시장모델에 기초한 시그널 모듈.
CSignalITF	시간별 시그널 필터링 모듈.
CSignalMACD	오실레이터 MACD의 시장 모델에 기반한 시그널 모듈.
CSignalMA	이동평균지표의 시장모델에 기초한 시그널 모듈.
CSignalSAR	지표 포물선 SAR의 시장모델에 기초한 시그널 모듈.
CSignalRSI	오실레이터 상대강도지수 시장모델에 기초한 시그널 모듈.
CSignalRVI	오실레이터 상대 활력 지수 시장 모델에 기반한 시그널 모듈.
CSignalStoch	진동자 확률론적 시장모델에 기초한 시그널 모듈.
CSignalTRIX	진동자 확률론적 시장모델에 기초한 시그널 모듈.

© 2000-2025, MetaQuotes Ltd.

Trading signal classes	Description
CSignalTEMA	3중 지수 이동평균 지표의 시장모델에 기초한 시그널 모듈.
CSignalWPR	오실레이터 윌리엄스 백분율 범위의 시장 모델에 기초한 시그널 모듈.

추적 손절매 클래스	Description
CTrailingFixedPips	이 클래스는 고정점을 기반으로 추적 손절매 알고리즘을 구현합니다
CTrailingMA	이 클래스는 이동 평균 지표 값을 기반으로 추적 손절매 알고리즘을 구현합니다
CTrailingNone	스텁 클래스, 추적 손절매 알고리즘을 사용하지 않음
CTrailingPSAR	이 클래스는 파라볼릭 SAR 지표 값을 기반으로 추적 손절매 알고리즘을 구현합니다

자금 관리 클래스	Description
CMoneyFixedLot	미리 정의된 고정 로트 크기를 사용한 거래를 기반으로 하는 알고리즘이 있는 클래스.
CMoneyFixedMargin	고정 마진이 미리 정의된 거래를 기반으로 하는 알고리즘이 있는 클래스.
CMoneyFixedRisk	리스크가 미리 정의된 거래를 기반으로 하는 알고리즘이 있는 클래스.
CMoneyNone	최소 허용 로트 크기를 가진 거래를 기반으로 하는 알고리즘이 있는 클래스.
CMoneySizeOptimized	알고리즘 변수 많은 크기로 거래에 기초해 이전 거래의 결과에 따라 결정하는 클래스.

© 2000-2025, MetaQuotes Ltd.

Expert Advisor용 기본 클래스

이 절에는 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명과 거래 전략의 작성 및 테스트를 위한 클래스 작업에 대한 기술적 세부 정보가 포함되어 있습니다.

이러한 클래스를 사용하면 거래 전략을 만들 때(특히 테스트) 시간을 절약할 수 있습니다.

MQL5 표준 라이브러리(트레이딩 전략 클래스 측면)는 터미널 디렉터리인 Include₩Expert 폴더에 배치됩니다.

Class	Description
CExpertBase	모든 거래 전략 클래스의 기본 클래스
CExpert	Expert Advisor용 기본 클래스
CExpertSignal	거래 시그널 클래스의 기본 클래스
CExpertTrailing	추적 손절매 클래스의 기본 클래스
CExpertMoney	자금 관리 클래스의 기본 클래스

© 2000-2025, MetaQuotes Ltd.

CExpertBase

CExpertBase는 CExpert 클래스 및 모든 보조 거래 전략 클래스의 기본 클래스입니다.

Description

CExpertBase는 Expert Advisor의 모든 개체에 공통적인 데이터와 메소드를 제공합니다.

Declaration

```
class CExpertBase : public CObject
```

Title

```
#include <Expert\ExpertBase.mqh>
```

상속 계층

CObject

CExpertBase

직계 하위

CExpert, CExpertMoney, CExpertSignal, CExpertTrailing

그룹별 클래스 메서드

Public Methods:

Initialization	
virtual Init	객체 초기화
virtual ValidationSettings	설정 확인
Parameters	
Symbol	심볼 설정
Period	타임프레임 설정
Magic	Expert Advisor ID 설정
지표 및 시계열	
virtual SetPriceSeries	외부 시계열에 대한 포인터 설정(가격 시리즈)
virtual SetOtherSeries	외부 시계열에 대한 포인터 설정(가격아닌 시리즈)
virtual InitIndicators	지표 및 시계열을 초기화합니다.
보호 데이터로 액세스	
InitPhase	현재 개체 초기화 단계를 가져옵니다
TrendType	추세 유형 설정

© 2000-2025, MetaQuotes Ltd.

Initialization	
UsedSeries	사용된 시계열의 비트마스크를 가져옵니다
EveryTick	"모든 틱" 플래그를 설정합니다
시계열에 액세스	
Open	인덱스별로 시가 시계열 요소를 가져옵니다
High	인덱스별로 고가 시계열 요소를 가져옵니다
Low	인덱스별로 저가 시계열 요소를 가져옵니다
Close	인덱스별로 종가 시계열 요소를 가져옵니다
Spread	인덱스별로 스프레드 시계열 요소를 가져옵니다
Time	인덱스별로 시간 시계열 요소를 가져옵니다
TickVolume	인덱스별로 TickVolume 시계열 요소를 가져옵니다
RealVolume	인덱스별로 RealVolume 시계열 요소를 가져옵니다

Protected Methods:

시계열 초기화	
InitOpen	시가 시계열 초기화 메서드
InitHigh	고가 시계열 초기화 메서드
InitLow	저가 시계열 초기화 메서드
InitClose	종가 시계열 초기화 메서드
InitSpread	스프레드 시계열 초기화 메서드
InitTime	시간 시계열 초기화 메서드
InitTickVolume	TickVolume 시계열 초기화 메서드
InitRealVolume	RealVolume 시계열 초기화 메서드
서비스 메서드	
virtual PriceLevelUnit	가격 수준 단위를 가져옵니다
virtual StartIndex	분석할 시작 막대 인덱스를 가져옵니다
virtual CompareMagic	Expert Advisor ID와 지정된 값을 비교합니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

InitPhase

객체 초기화의 현재 단계를 가져옵니다.

```
ENUM_INIT_PHASE  InitPhase()
```

반환 값

객체 초기화의 현재 단계.

참고

개체 초기화는 여러 단계로 구성됩니다:

1. 초기화 시작.

- 시작 - 구성자 종료 후
- 종료 - Init(...) 메서드의 성공적 완료 후.
- 허용 - Init(...) 메서드 호출
- 허용되지 않음 - ValidationSettings() 메서드 및 기차 초기화 메서드 호출

2. 파라미터 설정 단계. 이 단계에서는 지표를 만드는 데 사용되는 모든 개체 매개 변수를 설정해야 합니다.

- 시작 - Init(...) 메서드의 성공적 완료 후
- 종료 - ValidationSettings() 메서드의 성공적 완료 후
- 허용 - Symbol(...) 및 Period(...) 메서드 호출
- 허용되지 않음 - Init(...), SetPriceSeries(...), SetOtherSeries(...) 및 InitIndicators(...) 메서드 호출

3. 파라미터 검사.

- 시작 - ValidationSettings() 메서드의 성공적 완료 후
- 종료 - InitIndicators(...) 메서드의 성공적 완료 후
- 허용 - Symbol(...), Period(...) 및 InitIndicators(...) 메서드의 호출
- 허용되지 않음 - 다른 초기화 메서드 호출

4. 초기화 완료.

- 시작 - InitIndicators(...) 메서드의 성공적 완료 후
- 허용되지 않음 - 초기화 메서드 호출

© 2000-2025, MetaQuotes Ltd.

TrendType

추세 유형을 설정합니다.

```
void  TrendType(
   M_TYPE_TREND    value          // 값
   )
```

Parameters

value

[in] 추세 유형의 새 값.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

UsedSeries

사용된 시계열의 비트마스크를 가져옵니다.

```
int  UsedSeries()
```

반환 값

비트마스크로 사용된 시계열의 목록.

참고

비트가 설정된 경우 해당 시계열이 사용되고, 설정되지 않은 경우 시계열이 사용되지 않습니다.

해당 비트-시계열:

bit 0 - 시가 시계열,
bit 1 - 고가 시계열,
bit 2 - 저가 시계열,
bit 3 - 종가 시계열,
bit 4 - 스프레드 시계열,
bit 5 - 시간 시계열,
bit 6 - TickVolume 시계열,
bit 7 - RealVolume 시계열.

© 2000-2025, MetaQuotes Ltd.

EveryTick

"Every tick" 플래그를 설정합니다.

```
void  EveryTick(
   bool    value         // 플래그
   )
```

Parameters

value

[in] 플래그의 새 값.

반환 값

None.

참고

플래그가 설정되면 작업 심볼에서 각 가격 변경(틱)이 처리됩니다.

플래그가 설정되지 않은 경우 처리 방법은 작업 시간 및 심볼의 새 막대에서만 호출됩니다.

© 2000-2025, MetaQuotes Ltd.

Open

인덱스별로 시가 시계열 요소를 가져옵니다.

```
double  Open(
   int    ind         // 인덱스
   )
```

Parameters

ind

[in] 요소 인덱스.

반환 값

성공하면 지정된 인덱스로 시가 시계열 요소의 숫자 값을 반환하고, 그렇지 않으면 EMPTY_VALUE를 반환합니다.

참고

EMPTY_VALUE는 다음 두 가지 경우에 반환됩니다:

1. 시계열이가 사용되지 않음(해당 비트가 설정되지 않음).
2. 요소 인덱스가 범위 밖임.

© 2000-2025, MetaQuotes Ltd.

High

인덱스별로 고가 시계열 요소를 가져옵니다.

```
double  High(
   int    ind          // 인덱스
   )
```

Parameters

ind

[in] 요소 인덱스.

반환 값

성공하면 지정된 인덱스로 고가 시계열 요소의 숫자 값을 반환하고, 그렇지 않으면 EMPTY_VALUE를 반환합니다.

참고

EMPTY_VALUE는 다음 두 가지 경우에 반환됩니다:

1. 시계열이가 사용되지 않음(해당 비트가 설정되지 않음).
2. 요소 인덱스가 범위 밖임.

© 2000-2025, MetaQuotes Ltd.

Low

인덱스별로 저가 시간대의 요소를 가져옵니다.

```
double  Low(
   int     ind          // 인덱스
   )
```

Parameters

ind

[in] 요소 인덱스.

반환 값

성공하면 인덱스가 지정된 저가 시계열 요소의 숫자 값을 반환하고, 그렇지 않으면 EMPTY_VALUE를 반환합니다.

참고

EMPTY_VALUE는 다음 두 가지 경우에 반환됩니다:

1. 시계열이가 사용되지 않음(해당 비트가 설정되지 않음).
2. 요소 인덱스가 범위 밖임.

© 2000-2025, MetaQuotes Ltd.

Close

인덱스별로 종가 시계열 요소를 가져옵니다.

```
double  Close(
   int    ind         // 인덱스
   )
```

Parameters

ind

[in] 요소 인덱스.

반환 값

성공하면 지정된 인덱스와 함께 종가 시계열 요소의 숫자 값을 반환하고, 그렇지 않으면 EMPTY_VALUE를 반환합니다.

참고

EMPTY_VALUE는 다음 두 가지 경우에 반환됩니다:

1. 시계열이가 사용되지 않음(해당 비트가 설정되지 않음).
2. 요소 인덱스가 범위 밖임.

© 2000-2025, MetaQuotes Ltd.

Spread

인덱스별 스프레드 시계열의 요소를 가져옵니다.

```
double  Spread(
   int    ind         // 인덱스
   )
```

Parameters

ind

[in] 요소 인덱스.

반환 값

성공하면 지정된 인덱스와 함께 스프레드 시계열 요소의 숫자 값을 반환하고, 그렇지 않으면 EMPTY_VALUE를 반환합니다.

참고

EMPTY_VALUE는 다음 두 가지 경우에 반환됩니다:

1. 시계열이가 사용되지 않음(해당 비트가 설정되지 않음).
2. 요소 인덱스가 범위 밖임.

© 2000-2025, MetaQuotes Ltd.

Time

인덱스별로 시간 시계열의 요소를 가져옵니다.

```
datetime  Time(
   int    ind         // 인덱스
   )
```

Parameters

ind

[in] 요소 인덱스.

반환 값

성공하면 지정된 인덱스와 함께 Time timeseries 요소의 숫자 값을 반환하고, 그렇지 않으면 EMPTY_VALUE를 반환합니다.

참고

EMPTY_VALUE는 다음 두 가지 경우에 반환됩니다:

1. 시계열이가 사용되지 않음(해당 비트가 설정되지 않음).
2. 요소 인덱스가 범위 밖임.

© 2000-2025, MetaQuotes Ltd.

TickVolume

인덱스별로 TickVolume 시계열의 요소를 가져옵니다.

```
long  TickVolume(
   int    ind         // 인덱스
   )
```

Parameters

ind

[in] 요소 인덱스.

반환 값

성공하면 지정된 인덱스와 함께 TickVolume timeseries 요소의 숫자 값을 반환하고, 그렇지 않으면 EMPTY_VALUE를 반환합니다.

참고

EMPTY_VALUE는 다음 두 가지 경우에 반환됩니다:

1. 시계열이가 사용되지 않음(해당 비트가 설정되지 않음).
2. 요소 인덱스가 범위 밖임.

© 2000-2025, MetaQuotes Ltd.

RealVolume

인덱스별로 RealVolume 시계열 요소를 가져옵니다.

```
long  RealVolume(
   int    ind         // 인덱스
   )
```

Parameters

ind

[in] 요소 인덱스.

반환 값

성공하면 지정된 인덱스와 함께 RealVolume 시계열 요소의 숫자 값을 반환하고, 그렇지 않으면 EMPTY_VALUE를 반환합니다.

참고

EMPTY_VALUE는 다음 두 가지 경우에 반환됩니다:

1. 시계열이가 사용되지 않음(해당 비트가 설정되지 않음).
2. 요소 인덱스가 범위 밖임.

© 2000-2025, MetaQuotes Ltd.

Init

개체를 초기화합니다.

```
bool  Init(
   CSymbolInfo       symbol,   // 심볼
   ENUM_TIMEFRAMES   period,   // 타임프레임
   double            point     // 포인트
   )
```

Parameters

symbol

[in] 심볼 정보에 대한 액세스를 위해 CSymbolInfo 유형의 개체를 가리키는 포인터.

period

[in] 타임프레임 (ENUM_TIMEFRAMES 열거값).

point

[in] 2/4자리 포인트의 "가중치".

반환 값

성공적으로 완료되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Symbol

개체 심볼을 설정합니다.

```
bool  Symbol(
   string    name     // 심볼
   )
```

Parameters

name

[in] 심볼.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

개체가 첫 번째 초기화에 정의된 심볼과 다른 심볼을 사용하는 경우 작업 심볼을 설정해야 합니다.

© 2000-2025, MetaQuotes Ltd.

Period

객체 타임프레임을 설정합니다.

```
bool  Period(
   ENUM_TIMEFRAMES  value     // 타임프레임
   )
```

Parameters

value

[in] 타임프레임.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

객체가 초기화에 정의된 타임프레임과 다른 타임프레임을 사용하는 경우 작업 시간 설정이 필요합니다.

© 2000-2025, MetaQuotes Ltd.

Magic

Expert Advisor IDを設定します.

```
void  Magic(
   ulong  value     // 매직
   )
```

Parameters

value

[in] Expert Advisor ID.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

SetPriceSeries

외부 가격 시리즈에 대한 포인터를 설정합니다.

```
virtual bool  SetPriceSeries(
   CiOpen*    open,         // 포인터
   CiHigh*    high,         // 포인터
   CiLow*     low,          // 포인터
   CiClose*   close         // 포인터
   )
```

Parameters

open

[in] 시가 시계열에 대한 포인터.

high

[in] 고가 시계열에 대한 포인터.

low

[in] 저가 시계열에 대한 포인터.

close

[in] 종가 시계열에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

객체가 추가 작업에 필요한 심볼과 시간(첫 번째 초기화 중 설정) 및 가격 시간대를 사용하는 경우 외부 시계열(가격 시리즈)에 대한 포인터 설정이 필요합니다.

© 2000-2025, MetaQuotes Ltd.

SetOtherSeries

외부 비가격 시리즈에 대한 포인터를 설정합니다.

```
virtual bool  SetOtherSeries(
   CiSpread*       spread,        // 포인터
   CiTime*         time,          // 포인터
   CiTickVolume*   tick_volume,   // 포인터
   CiRealVolume*   real_volume    // 포인터
   )
```

Parameters

spread

[in] 스프레드 시계열 포인터.

time

[in] 시간 시계열 포인터.

tick_volume

[in] TickVolume 시계열에 대한 포인터.

real_volume

[in] RealVolume 시계열에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

객체가 추가 작업에 필요한 심볼과 시간(첫 번째 초기화 중 설정) 및 가격 시간대를 사용하는 경우 외부 시계열(가격 시리즈)에 대한 포인터 설정이 필요합니다.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

모든 지표와 시계열을 초기화합니다.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

시계열은 개체가 초기화할 때 정의된 심볼 또는 타임프레임과 다른 심볼 또는 타임프레임을 사용하는 경우에만 초기화됩니다.

© 2000-2025, MetaQuotes Ltd.

InitOpen

시가 시계열을 초기화합니다.

```
bool  InitOpen(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

시가 시계열은 객체가 첫 번째 초기화에 정의된 심볼/타임프레임과 다른 심볼/타임프레임을 사용하는 경우에만 초기화됩니다(그리고 타임레임이 추가로 사용됨).

© 2000-2025, MetaQuotes Ltd.

InitHigh

고가 시계열을 초기화합니다.

```
bool  InitHigh(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

고가 시계열은 객체가 첫 번째 초기화에 정의된 심볼/타임프레임과 다른 심볼/타임프레임을 사용하는 경우에만 초기화됩니다(그리고 타임프레임을 추가로 사용함).

© 2000-2025, MetaQuotes Ltd.

InitLow

저가 시계열을 초기화합니다

```
bool  InitLow(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

저가 시계열은 객체가 첫 번째 초기화에 정의된 기호/타임프레임과 다른 심볼/타임프레임을 사용하는 경우에만 초기화됩니다(그리고 타임레일이 추가로 사용됨).

© 2000-2025, MetaQuotes Ltd.

InitClose

종가 시계열을 초기화합니다.

```
bool  InitClose(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

종가 시계열은 Expert Advisor가 첫 번째 초기화에 정의된 심볼/타임프레임과 다른 심볼/타임프레임을 사용하는 경우에만 초기화됩니다.

© 2000-2025, MetaQuotes Ltd.

InitSpread

스프레드 시계열을 초기화합니다.

```
bool  InitSpread(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

스프레드 시계열은 Expert Advisor가 첫 번째 초기화(그리고 시계열이 더 사용됩니다)에 정의된 심볼/타임프레임과 다른 기호/타임프레임을 사용하는 경우에만 초기화됩니다.

© 2000-2025, MetaQuotes Ltd.

InitTime

시간 시계열을 초기화합니다.

```
bool  InitTime(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

시간 시계열은 Expert Advisor가 첫 번째 초기화에 정의된 심볼/타임프레임과 다른 심볼/타임프레임을 사용하는 경우에만 초기화됩니다.

© 2000-2025, MetaQuotes Ltd.

InitTickVolume

TickVolume 시계열을 초기화합니다.

```
bool  InitTickVolume(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

TickVolume 시계열은 Expert Advisor가 첫 번째 초기화(그리고 시계열이 더 사용됨)에 정의된 심볼/타임프레임과 다른 심볼/타임프레임을 사용하는 경우에만 초기화됩니다.

© 2000-2025, MetaQuotes Ltd.

InitRealVolume

RealVolume 시계열을 초기화합니다.

```
bool  InitRealVolume(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

RealVolume 시계열은 Expert Advisor가 첫 번째 초기화에 정의된 심볼/타임프레임과 다른 심볼/타임프레임을 사용하는 경우에만 초기화됩니다(그리고 시계열은 추가로 사용됩니다).

© 2000-2025, MetaQuotes Ltd.

PriceLevelUnit

가격 수준 단위를 가져옵니다.

```
virtual double  PriceLevelUnit()
```

반환 값

가격 수준 단위의 값.

참고

기본 클래스의 메서드는 2/4자리 포인트의 "가중치"를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

StartIndex

분석할 시작 표시줄 인덱스를 가져옵니다.

```
virtual int  StartIndex()
```

반환 값

분석할 시작 표시줄 인덱스.

참고

현재 막대를 분석할 플래그가 true(현재 막대의 분석)로 설정된 경우 메서드는 0을 반환합니다. 플래그가 설정되지 않은 경우 1을 반환합니다(마지막으로 완료된 막대에서 분석).

© 2000-2025, MetaQuotes Ltd.

CompareMagic

Expert Advisor ID(매직)를 지정된 값과 비교합니다.

```
virtual bool  CompareMagic(
   ulong  magic    // 식별자
   )
```

Parameters

magic

[in] 비교할 식별자 값.

반환 값

식별자가 특정 값과 일치시 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CExpert

CExpert는 거래 전략의 기본 클래스입니다.

그것은 이미 기본적인 "기술"을 가지고 있습니다. 시계열 및 지표 작업을 위한 알고리즘과 거래 전략을 위한 일련의 가상 방법을 내장하고 있습니다.

사용 방법:

1. 전략의 알고리즘 준비;
2. CExpert 클래스에서 상속된 자신만의 클래스 만들기;
3. 클래스의 가상 메서드를 사용자 알고리즘으로 재정의합니다.

Description

CExpert 클래스는 거래 전략의 구현을 위한 가상 방법 집합입니다.

참고

특성 m_symbol 및 m_magic 쌍을 기준으로 Expert Advisor의 직위로 인식되어 관리됩니다. "헤징" 모드에서는 동일한 기호에 대해 여러 포지션을 오픈할 수 있으므로 m_magic 값이 중요합니다.

Declaration

```
class CExpert : public CExpertBase
```

Title

```
#include <Expert\Expert.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpert

그룹별 클래스 메서드

Initialization	
Init	클래스 인스턴스 초기화 방법
virtual InitSignal	거래 시그널 개체를 초기화합니다
virtual InitTrailing	추적 손절매 개체 초기화
virtual InitMoney	자금 관리 객체 초기화
virtual InitTrade	거래 객체 초기화
virtual ValidationSettings	설정 확인
virtual InitIndicators	지표 및 시간대 초기화

© 2000-2025, MetaQuotes Ltd.

Initialization	
virtual InitParameters	Expert Advisor 매개 변수 초기화 방법
virtual Deinit	클래스 인스턴스 초기화 해제 방법
virtual DeinitSignal	거래 시그널 개체 초기화 취소
virtual DeinitTrailing	추적 손절매 개체 초기화 취소
virtual DeinitMoney	자금 관리 개체 초기화 취소
virtual DeinitTrade	거래 객체 초기화 취소
virtual DeinitIndicators	지표 및 시계열 초기화 취소
Parameters	
Magic	Expert Advisor ID 설정
MaxOrders	허용된 주문의 최대 양 가져오기/설정
OnTickProcess	"OnTick" 이벤트를 진행하도록 플래그를 설정합니다
OnTradeProcess	"OnTrade" 이벤트를 진행하도록 플래그를 설정합니다.
OnTimerProcess	"OnTimer" 이벤트를 진행하도록 플래그를 설정합니다
OnChartEventProcess	"OnChartEvent" 이벤트를 진행하도록 플래그를 설정합니다
OnBookEventProcess	"OnChartEvent" 이벤트를 진행하도록 플래그를 설정합니다
이벤트 처리 방법	
OnTick	OnTick 이벤트 핸들러
OnTrade	OnTrade 이벤트 핸들러
OnTimer	OnTimer 이벤트 핸들러
OnChartEvent	OnChartEvent 이벤트 핸들러
OnBookEvent	OnBookEvent 이벤트 핸들러
메서드 업데이트	
Refresh	모든 정보 업데이트
Processing	
Processing	주 처리 알고리즘
시장 진입 메서드	
CheckOpen	포지션 오프닝 상태를 점검합니다
CheckOpenLong	롱 포지션 오픈 상태를 점검합니다
CheckOpenShort	숏 포지션 오픈 조건을 점검합니다
OpenLong	롱 포지션 오픈

© 2000-2025, MetaQuotes Ltd.

Initialization	
OpenShort	숏 포지션 오픈
시장 엑싯 메서드	
CheckClose	현재 포지션을 클로즈하기 위한 조건을 점검합니다
CheckCloseLong	롱 포지션을 클로즈하기 위한 조건을 점검합니다
CheckCloseShort	숏 포지션을 클로즈하기 위한 조건을 점검합니다
CloseAll	오픈 포지션을 클로즈하고 모든 주문을 삭제합니다
Close	오픈 포지션을 클로즈합니다
CloseLong	롱 포지션을 클로즈합니다
CloseShort	숏 포지션을 클로즈합니다
포지션 리버스 메서드	
CheckReverse	오픈 포지션 리버스에 대한 조건을 점검합니다
CheckReverseLong	롱 포지션 리버스에 대한 조건을 점검합니다.
CheckReverseShort	숏 포지션 반전에 대한 조건을 점검합니다
ReverseLong	롱 포지션의 리버스 작업을 수행합니다
ReverseShort	숏 포지션의 리버스 작동을 수행합니다
포지션/주문 추적 메서드법	
CheckTrailingStop	포지션 매개 변수를 수정하기 위한 조건을 확인합니다
CheckTrailingStopLong	롱 포지션 추적 손절매 상태 점검
CheckTrailingStopShort	숏 포지션 추적 손절매 상태 점검
TrailingStopLong	롱 포지션 추적 손절매 수행
TrailingStopShort	숏 포지션 추절 손절매 수행
CheckTrailingOrderLong	스탑/스탑 매수 주문 추적 손절매 조건 점검
CheckTrailingOrderShort	리밋/스탑 매도 주문 추적 손절매 조건 점검
TrailingOrderLong	리밋/스탑 매수 주문 추적 손절매 수행
TrailingOrderShort	리밋/스탑 매도 주문 추적 손절매 수행
주문 삭제 메서드	
CheckDeleteOrderLong	매수 주문 삭제 조건 확인
CheckDeleteOrderShort	매도 주문 삭제 조건 확인
DeleteOrders	모든 주문 삭제
DeleteOrder	스탑/리밋 주문 삭제

© 2000-2025, MetaQuotes Ltd.

Initialization	
DeleteOrderLong	리밋/스탑 매수 주문 삭제
DeleteOrderShort	리밋/스탑 매도 주문 삭제
거래량 메서드	
LotOpenLong	매수 작업에 대한 거래량을 가져옵니다
LotOpenShort	매도 작업에 대한 거래량을 가져옵니다
LotReverse	포지션 리버스 작업에 대한 거래량을 가져옵니다
거래내역 메서드	
PrepareHistoryDate	거래 내역 추적의 시작 날짜를 설정합니다
HistoryPoint	거래 내역의 체크포인트 생성(직급 수, 주문 수, 거래 수 및 과거 주문 저장)
CheckTradeState	현재 상태를 저장된 상태와 비교하고 해당 이벤트 핸들러를 호출합니다
이벤트 플래그	
WaitEvent	거래 이벤트 대기 플래그를 설정합니다
NoWaitEvent	거래 이벤트 대기 플래그 재설정
거래 이벤트 처리 메서드	
TradeEventPositionStopTake	"포지션 스탑 로스 / 이익실현 트리거" 이벤트의 이벤트 핸들러
TradeEventOrderTriggered	"지정가 주문 트리거" 이벤트의 이벤트 핸들러
TradeEventPositionOpened	"오픈된 포지션" 이벤트의 이벤트 핸들러
TradeEventPositionVolumeChanged	오픈 포지션 이벤트의 이벤트 핸들러
TradeEventPositionModified	"수정된 포지션" 이벤트의 이벤트 핸들러
TradeEventPositionClosed	"포지션 클로즈" 이벤트의 이벤트 핸들러
TradeEventOrderPlaced	"지정가 주문 실행" 이벤트의 이벤트 핸들러
TradeEventOrderModified	"지정가 주문 수정" 이벤트의 이벤트 핸들러
TradeEventOrderDeleted	"지정가 주문 삭제" 이벤트의 이벤트 핸들러
TradeEventNotIdentified	식별되지 않은 이벤트의 이벤트 핸들러
서비스 메서드	
TimeframeAdd	추적할 타임프레임을 추가합니다
TimeframesFlags	타임프레임 플래그를 형성합니다
SelectPosition	작업할 포지션을 선택합니다

© 2000-2025, MetaQuotes Ltd.

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

Init

클래스 인스턴스 초기화 메서드.

```
bool  Init(
   string              symbol,         // 심볼
   ENUM_TIMEFRAMES     period,         // 타임프레임
   bool                every_tick,     // 플래그
   ulong               magic           // Expert Advisor 식별자
   )
```

Parameters

symbol

[in] 심볼.

period

[in] ENUM_TIMEFRAMES 열거값으로부터의 타임프레임.

every_tick

[in] 플래그.

magic

[in] Expert Advisor ID (매직 넘버).

반환 값

None.

참고

every_tick이 true로 설정된 경우 작업 심볼의 각 틱에서 Processing() 메서드가 호출됩니다. 그렇지 않으면, Processing()는 EA의 작업 심볼 작업 타임프레임에 새로운 막대가 형성될 때만 호출됩니다.

© 2000-2025, MetaQuotes Ltd.

Magic

Expert Advisor ID를 설정합니다(magic).

```
void  Magic(
   ulong  value     // 새 값
   )
```

Parameters

value

[in] Expert Advisor ID의 새 값.

반환 값

None.

참고

EA의 식별자가 변경되면 모든 보조 객체에 동일한 식별자가 할당됩니다.

Implementation

```
//+------------------------------------------------------------------+
//| 객체와 그 종속(자녀) 객체의 매직 넘버를 설정합니다                |
//| INPUT:  value - 매직 넘버의 새 값.                               |
//| OUTPUT: no.                                                      |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
void CExpert::Magic(ulong value)
  {
   if(m_trade!=NULL)     m_trade.SetExpertMagicNumber(value);
   if(m_signal!=NULL)    m_signal.Magic(value);
   if(m_money!=NULL)     m_money.Magic(value);
   if(m_trailing!=NULL) m_trailing.Magic(value);
//---
   CExpertBase::Magic(value);
  }
```

© 2000-2025, MetaQuotes Ltd.

InitSignal

거래 시그널 객체를 초기화합니다.

```
virtual bool  InitSignal(
   CExpertSignal*     signal=NULL,        // 포인터
   )
```

Parameters

signal

[in] CExpertSignal 클래스 객체 (또는 자녀객체)에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

시그널이 NULL이면, CExpertSignal 클래스는 초기화에 사용됩니다(아무것도 하지않습니다).

© 2000-2025, MetaQuotes Ltd.

InitTrailing

츠작 손절매 객체를 초기화합니다.

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,        // 포인터
   )
```

Parameters

trailing

[in] CExpertTrailing 클래스 객체(또는 이의 자녀객체)에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

추적이 NULL이면, ExpertTrailing 클래스는 초기화에 사용됩니다(아무것도 하지않습니다).

© 2000-2025, MetaQuotes Ltd.

InitMoney

자금 관리 개체를 초기화합니다.

```
virtual bool  InitMoney(
   CExpertMoney*     money=NULL,         // 포인터
   )
```

Parameters

money

[in] CExpertMoney클래스 객체(또는 그것의 자녀객체)에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

참고

자금이 NULL이면, CExpertMoney 클래스는 초기화에 사용될 것입니다(최소 로트 사용).

© 2000-2025, MetaQuotes Ltd.

InitTrade

거래 객체를 초기화합니다.

```
virtual bool  InitTrade(
   ulong            magic,       // 식별자
   CExpertTrade*    trade=NULL   // 포인터
   )
```

Parameters

magic

[in] Expert Advisor ID (거래 요청에 사용됨).

trade

[in] 거래 객체에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Deinit

클래스 인스턴스 초기화 해제 메서드.

```
virtual void Deinit()
```

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

OnTickProcess

OnTick 이벤트 핸들러 플래그를 설정합니다.

```
void  OnTickOProcess(
   bool     value        // 플래그
   )
```

Parameters

value

[in] OnTick 이벤트 핸들링 플래그.

반환 값

None.

참고

플래그가 참이면 OnTick 이벤트가 처리됩니다. 기본적으로 플래그가 true로 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

OnTradeProcess

OnTrade 이벤트 핸들링 플래그를 설정합니다.

```
void  OnTradeProcess(
   bool     value        // 플래그
   )
```

Parameters

value

[in] OnTrade 이벤트 핸들링 플래그.

반환 값

None.

참고

플래그가 참이면, OnTrade 이벤트가 처리됩니다. 기본적으로 플래그는 false로 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

OnTimerProcess

OnTimer 이벤트 핸들링 플래그를 설정합니다.

```
void  OnTimerProcess(
   bool     value        // 플래그
   )
```

Parameters

value

[in] OnTimer 이벤트 핸들링 플래그.

반환 값

None.

참고

플래그가 참이면, OnTimer 이벤트가 처리됩니다. 기본적으로 플래그는 false로 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

OnChartEventProcess

OnChartEvent 이벤트를 처리할 플래그를 설정합니다.

```
void  OnChartEventProcess(
   bool     value        // 플래그
   )
```

Parameters

value

[in] OnChartEvent 이벤트를 처리할 플래그를 설정합니다.

반환 값

None.

참고

플래그가 참이면, OnChartEvent 이벤트가 처리됩니다. 기본적으로 플래그는 false로 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

OnBookEventProcess

OnBookEvent 이벤트를 처리할 플래그를 설정합니다.

```
void  OnChartEventProcess(
   bool     value        // 플래그
   )
```

Parameters

value

[in] OnBookEvent 이벤트를 처리할 플래그.

반환 값

None.

참고

플래그가 참이면 OnBookEvent 이벤트가 처리됩니다. 기본적으로 플래그는 false로 설정됩니다.

© 2000-2025, MetaQuotes Ltd.

MaxOrders (Get Method)

허용된 주문의 최대 양을 가져옵니다.

```
int  MaxOrders()
```

반환 값

허용된 주문의 최대 양.

MaxOrders (Set Method)

허용된 주문의 최대 양을 설정합니다.

```
void  MaxOrders(
   int     max_orders        // 주문량
   )
```

Parameters

max_orders

[in] 허용된 주문의 최대 금액의 새 값.

반환 값

None.

참고

기본적으로 허용되는 주문의 최대 양은 1입니다.

© 2000-2025, MetaQuotes Ltd.

Signal

トレイドド シグネル 개체에 대한 포인터를 가져옵니다.

```
CExpertSignal*  Signal() const
```

반환 값

트레이드 시그널 객체에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

성공이면 true, 그렇지 않으면 false.

참고

또한 모든 Expert Advisor 개체의 설정도 확인합니다.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

필요한 지표 및 시계열을 초기화합니다.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공적으로 완료되면 true, 그렇지 않으면 false.

참고

객체가 초기화에 정의된 것 외의 심볼 또는 타임프레임을 사용하는 경우 시계열은 초기화됩니다.

모든 보조 EA 객체의 지표와 시계열이 초기화됩니다.

© 2000-2025, MetaQuotes Ltd.

OnTick

OnTick 이벤트 핸들러.

```
virtual void  OnTick()
```

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

OnTrade

OnTrade 이벤트 핸들러.

```
virtual void  OnTrade()
```

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

OnTimer

OnTimer 이벤트 핸들러.

```
virtual void  OnTimer()
```

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

OnChartEvent

OnChartEvent 이벤트 핸들러.

```
virtual void  OnChartEvent(
   const int     id,       // 이벤트 id
   const long&   lparam,   // long 타입의 이벤트 파라미터
   const double  dparam,   // double 타입의 이벤트 파라미터
   const string  sparam    // string 타입의 이벤트 파라미터
   )
```

Parameters

id

[in] Event ID.

lparam

[in] long 타입의 이벤트 파라미터.

dparam

[in] double 타입의 이벤트 파라미터.

sparam

[in] string 타입의 이벤트 파라미터.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

OnBookEvent

OnBookEvent 이벤트 핸들러.

```
virtual void  OnBookEvent(
   const string&     symbol       // 심볼
)
```

Parameters

symbol

[in] OnBookEvent 이벤트의 심볼.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

InitParameters

Expert Advisorの パラメータを 초기화합니다.

```
virtual bool  InitParameters()
```

반환 값

성공이면 true, 그렇지 않으면 false.

참고

CExpert 기본 클래스의 InitParameters() 함수는 아무 것도 하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

DeinitTrade

거래 객체를 초기화 해제합니다.

```
virtual void  DeinitTrade()
```

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

DeinitSignal

거래 시그널 객체를 초기화 해제합니다.

```
virtual void  DeinitSignal()
```

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

DeinitTrailing

추적 객체를 초기화 해제합니다.

```
virtual void  DeinitTrailing()
```

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

DeinitMoney

자금 관리 개체를 초기화 해제합니다.

```
virtual void  DeinitMoney()
```

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

DeinitIndicators

지표 및 시계열을 초기화 해제합니다.

```
virtual void  DeinitIndicators()
```

반환 값

None.

참고

모든 보조 EA 객체의 지표와 시계열은 초기화 해제됩니다.

© 2000-2025, MetaQuotes Ltd.

Refresh

필요한 모든 데이터를 업데이트합니다.

```
virtual bool  Refresh()
```

반환 값

추가 틱 처리가 필요하면 true, 그렇지 않으면 false.

참고

이를 통해 틱 처리의 필요성을 결정할 수 있습니다. 필요한 경우 모든 견적, 시계열 및 지표 데이터를 업데이트하고 true를 반환합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 처리할 데이터 갱신                                               |
//| INPUT:  no.                                                      |
//| OUTPUT: 성공적이면 true, 그렇지 않으면 false.                    |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::Refresh()
  {
   MqlDateTime time;
//--- 요율 갱신
   if(!m_symbol.RefreshRates()) return(false);
//--- 처리 필요성 확인
   TimeToStruct(m_symbol.Time(),time);
   if(m_period_flags!=WRONG_VALUE && m_period_flags!=0)
      if((m_period_flags & TimeframesFlags(time))==0) return(false);
   m_last_tick_time=time;
//--- 지표 갱신
   m_indicators.Refresh();
//--- ok
   return(true);
  }
```

© 2000-2025, MetaQuotes Ltd.

Processing

주 처리 알고리즘.

```
virtual bool  Processing()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

다음 단계를 수행합니다:

1. 심볼에 오픈된 포지션이 있는지 확인합니다. 오픈된 포지션이 없으면 2, 3, 4단계를 건너뜁니다.
2. 오픈 포지션 리버스에 대한 조건을 확인합니다(CheckReverse() 메서드). 포지션이 "리버스"된 경우 종료합니다.
3. 포지션을 클로즈 하기위한 조건을 확인합니다(CheckClose() 메서드). 포지션이 클로즈된 경우 4단계를 건너뜁니다.
4. 포지션 매개 변수를 수정하기 위한 조건을 확인합니다(CheckTrailingStop() 메서드). 포지션 매개 변수가 수정된 경우 종료합니다.
5. 심볼에 지정가 주문이 있는지 확인합니다. 지정가 주문이 없으면 9단계로 이동합니다.
6. 주문을 삭제할 조건을 확인합니다(지정가 매수 주문의 경우 CheckDeleteOrderLong(), 지정가 매도 주문의 경우 CheckDeleteOrderShort()). 주문이 삭제된 경우 9단계로 이동합니다.
7. 지정가 주문 매개 변수를 수정하려면 조건을 선택합니다(매수 주문의 경우 CheckTrailingOrderLong() 또는 매도 주문의 경우 CheckTrailingOrderShort()). 주문 매개 변수가 수정되었으면 종료합니다.
8. Exit.
9. 포지션 오픈에 대한 조건을 확인합니다(CheckOpen() 메서드).

고유한 알고리즘을 구현하려면 하위 클래스의 Processing() 메서드를 재정의해야 합니다.

Implementation

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 주 함수                                                          |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 처리되면 true, 그렇지 않으면 false.  |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::Processing()
  {
//--- 포지션을 오픈할 수 있는지 확인
   if(m_position.Select(m_symbol.Name()))
     {
      //--- 오픈 포지션 사용가능
      //--- 포지션 리버스 가능성을 확인
      if(CheckReverse()) return(true);
      //--- 포지션 클로징/지정가 주문 삭제 가능성 확인
      if(!CheckClose())
        {
         //--- 포지션 수정 가능성 확인
         if(CheckTrailingStop()) return(true);
         //--- 작업 없이 반환됩니다
         return(false);
        }
     }
//--- 지정가 주문 실행 확인
   int total=OrdersTotal();
   if(total!=0)
     {
      for(int i=total-1;i>=0;i--)
        {
         m_order.SelectByIndex(i);
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         if(m_order.OrderType()==ORDER_TYPE_BUY_LIMIT || m_order.OrderType()==ORDER_TY
           {
            //--- 지정가 주문 삭제 기능 확인
            if(CheckDeleteOrderLong()) return(true);
            //--- 지정가 주문 매수 가능성 확인
            if(CheckTrailingOrderLong()) return(true);
           }
         else
           {
            //--- 판매 대기중 주문 삭제 능력 확인
            if(CheckDeleteOrderShort()) return(true);
            //--- 지정가 주문 수정 가능성 확인
            if(CheckTrailingOrderShort()) return(true);
           }
         //--- 작업 없이 반환합니다
         return(false);
        }
     }
//--- 포지션 오픈/지정가 주문 설정 가능성 확인
   if(CheckOpen()) return(true);
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

SelectPosition

작업할 포지션을 선택합니다.

```
void  SelectPosition()
```

반환 값

No.

Implementation

```
//+------------------------------------------------------------------+
//| 포지션 선택                                                      |
//| INPUT:  no.                                                      |
//| OUTPUT: no.                                                      |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::SelectPosition(void)
  {
   bool res=false;
//---
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      res=m_position.SelectByMagic(m_symbol.Name(),m_magic);
   else
      res=m_position.Select(m_symbol.Name());
//---
   return(res);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpen

포지션 오픈을 위한 조건을 확인합니다.

```
virtual bool  CheckOpen()
```

반환 값

거래 작업이 실행되었으면 true, 아니면 false.

참고

롱(CheckOpenLong()) 및 숏(CheckOpenShort()) 포지션 오픈 필요성을 확인합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 포지션 오픈 또는 리밋/스탑 주문 설정을 확인합니다                  |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckOpen()
  {
   if(CheckOpenLong())  return(true);
   if(CheckOpenShort()) return(true);
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

롱 포지션을 오픈하기 위한 필요성과 조건을 확인합니다.

```
virtual bool  CheckOpenLong()
```

반환 값

거래 작업이 실행되었으면 true, 아니면 false.

참고

롱 포지션(시그널 객체의 CheckOpenLong() 메서드)를 오픈할 필요가 있는지 확인하고 조건이 충족될 경우 시그널 객체(OpenLong() 메서드)에서 설정한 파라미터로 오픈할 수 있습니다.

Implementation

```
//+------------------------------------------------------------------+
//| 롱 포지션 오픈 또는 리밋/스탑 주문 설정을 확인합니다              |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- 롱 입력 작업을 위한 시그널을 확인합니다
   if(m_signal.CheckOpenLong(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenLong(price,sl,tp));
     }
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

필요성과 조건을 확인하여 숏 포지션을 오픈합니다.

```
virtual bool  CheckOpenShort()
```

반환 값

거래 작업이 실행되었으면 true, 아니면 false.

참고

롱 포지션(시그널 객체의의 CheckOpenShort() 메서드)를 오픈할 필요가 있는지 확인하고 조건이 충족되면 시그널 객체(OpenShort() 메서드)에서 설정한 파라미터로 이 작업을 수행합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 숏 포지션 오픈 또는 리밋/스탑 주문 설정을 확인합니다             |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- 숏 입력 작업을 위한 시그널을 확인합니다
   if(m_signal.CheckOpenShort(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenShort(price,sl,tp));
     }
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

OpenLong

롱 포지션을 오픈합니다.

```
virtual bool  OpenLong(
   double    price,    // 가격
   double    sl,       // 스탑 로스
   double    tp        // 이익 실현
   )
```

Parameters

price

[in] 시장 진입가.

sl

[in] 스탑 로스 가격.

tp

[in] 이익 실현 가격.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

거래량(LotOpenLong(...) 메서드)을 가져오고 거래량이 0이 아닌 경우 롱 포지션(Buy() 거래 객체 메서드)을 오픈합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 롱 포지션 오픈 또는 리밋/스탑 주문 설정                          |
//| INPUT:  price - 가격,                                            |
//|         sl    - 스탑 로스,                                       |
//|         tp    - 이익 실현.                                       |
//| OUTPUT: 거래 작업이 처리되면 true, 그렇지 않으면 false.          |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::OpenLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- 오픈용 로트를 가져옵니다
   double lot=LotOpenLong(price,sl);
//--- 오픈용 로트를 확인합니다
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

OpenShort

숏 포지션을 오픈합니다.

```
virtual bool  OpenShort(
   double    price,    // 가격
   double    sl,       // 스탑 로스
   double    tp        // 이익 실현
   )
```

Parameters

price

[in] 시장 진입가.

sl

[in] 스탑 로스 가격.

tp

[in] 이익 실현 가격.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

거래량(LotOpenShort() 메서드)을 가져오고 거래량이 0이 아닌 경우 숏 포지션(거래 객체 매도 메서드를 호출하여)을 오픈합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 숏 포지션 오픈 또는 리밋/스탑 주문 설정                          |
//| INPUT:  price - 가격,                                            |
//|         sl    - 스탑 로스,                                       |
//|         tp    - 이익 실현.                                       |
//| OUTPUT: 거래 작업이 성공적이면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::OpenShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- 오픈용 로트 가져오기
   double lot=LotOpenShort(price,sl);
//--- 오픈용 로트 확인하기
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverse

오픈 포지션을 리버스하기 위한 필요성 및 조건을 점검합니다.

```
virtual bool  CheckReverse()
```

반환 값

거래 작업이 실행되었으면 true, 아니면 false.

참고

이것은 롱(CheckReverseLong()) 및 숏(CheckReverseShort()) 포지션의 리버스 필요성을 확인합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 포지션 리버스를 확인합니다                                       |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckReverse()
  {
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- 롱 포지션의 리버스 가능성을 확인합니다
      if(CheckReverseLong())  return(true);
     }
   else
      //--- 숏 포지션 반전 가능성을 확인합니다
      if(CheckReverseShort()) return(true);
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverseLong

롱 포지션을 리버스하기 위한 필요성 및 조건을 확인합니다.

```
virtual bool  CheckReverseLong()
```

반환 값

거래 작업이 실행되었으면 true, 아니면 false.

참고

조건이 충족될 경우 시그널 객체((ReverseLong(...) 메서드)에서 설정한 파라미터로 롱 포지션(시그널 객체의 CheckReverseLong() 메서드)의 리버스 작업을 수행해야 할 필요성을 확인합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 롱 포지션 리버스를 확인합니다                                    |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- 시그널의 롱 리버스 작업을 확인합니다
   if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,t
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckReverseShort

필요성과 조건을 점검하여 롱 포지션을 리버스 합니다.

```
virtual bool  CheckReverseLong()
```

반환 값

거래 작업이 실행되었으면 true, 아니면 false.

참고

조건이 충족될 경우 시그널 객체(ReverseShort() 메서드)에서 설정한 파라미터로 숏 포지션(신호 객체의 CheckReverseShort() 메서드)의 리버스 작업을 수행해야 할 필요성을 확인합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 숏 포지션 리버스를 확인합니다                                    |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- 시그널에서 숏 리버스 작업을 확인합니다
   if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,sl
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

ReverseLong

롱 포지션의 리버스 작업을 수행합니다.

```
virtual bool  ReverseLong(
   double    price,    // 가격
   double    sl,       // 스탑 로스
   double    tp        // 이익 실현
   )
```

Parameters

price

[in] 시장 진입가격.

sl

[in] 스탑 로스 가격.

tp

[in] 이익 실현 가격.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

이것은 포지션 리버스 볼륨(LotReverse() 메서드)을 가져오고 거래량이 0이 아닌 경우 롱 포지션(Sell() 메서드)를 반전시킵니다.

포지션 계산의 "헤징" 모드에서 포지션 리버스는 기존 포지션을 클로즈하고 남은 볼륨으로 새로운 반전 포지션을 오픈할 때 수행됩니다.

Implementation

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 롱 포지션 리버스                                                 |
//| INPUT:  price - 가격,                                            |
//|         sl    - 스탑 로스,                                       |
//|         tp    - 이익 실현.                                       |
//| OUTPUT: 거래 작업이 처리되면 true, 그렇지 않으면 false.          |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE)
      return(false);
//--- 리버스용 로트 가져오기
   double lot=LotReverse(sl);
//--- 로트 확인하기
   if(lot==0.0)
      return(false);
//---
   bool result=true;
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
     {
      //--- first close existing position
      lot-=m_position.Volume();
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
     }
   if(result)
      result=m_trade.Sell(lot,price,sl,tp);
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

ReverseShort

숏 포지션의 리버스 작업을 수행합니다.

```
virtual bool  ReverseShort(
   double    price,    // 가격
   double    sl,       // 스탑 로스
   double    tp        // 이익 실현
   )
```

Parameters

price

[in] 시장 진입가.

sl

[in] 스탑 로스 가격.

tp

[in] 이익 실현 가격.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

거래량이 0이 아닐 경우 포지션 리버스(LotReverse(...) 메서드)을 가져오고 숏 포지션 반전(거래 객체의 Buy() 메서드) 트레이딩 연산을 수행합니다.

포지션 계산의 "헤징" 모드에서 포지션 리버스는 기존 포지션을 클로즈하고 남은 볼륨으로 새로운 반전 포지션을 오픈할 때 수행됩니다.

Implementation

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 숏 포지션 리버스                                                 |
//| INPUT:  price - 가격,                                            |
//|         sl    - 스탑 로스,                                       |
//|         tp    - 이익 실현.                                       |
//| OUTPUT: 거래 작업이 처리되면 true, 그렇지 않으면 false.          |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE)
      return(false);
//--- 리버스용 로트 가져오기
   double lot=LotReverse(sl);
//--- 로트 확인하기
   if(lot==0.0)
      return(false);
//---
   bool result=true;
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
     {
      //--- first close existing position
      lot-=m_position.Volume();
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
     }
   if(result)
      result=m_trade.Sell(lot,price,sl,tp);
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckClose

클로즈 포지션을 확인할 조건을 확인합니다.

```
virtual bool  CheckClose()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

1. 이것은 Expert Advisor Stop Out 조건(자금 관리 객체의 CheckClose() 메서드)을 확인합니다. 조건이 충족되면 포지션을 클로즈 하고 모든 주문 (CloseAll())을 삭제하고 종료합니다.
2. 롱 또는 숏 포지션(CheckCloseLong() 또는 CheckCloseShort() 메서드) 를 닫기 위한 조건을 확인하고, 포지션이 클로즈된 경우 모든 주문(DeleteOrders() 메서드)을 삭제합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 포지션 클로즈 또는 리밋/스탑 주문 삭제 여부를 확인하십시오              |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckClose()
  {
   double lot;
//--- 호출 전에 포지션을 선택해야 합니다
   if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
      return(CloseAll(lot));
//--- 포지션 유형을 확인합니다
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- 롱 포지션의 클로징 / 지정가 매수 주문 삭제 가능성을 확인합니다
      if(CheckCloseLong())
        {
         DeleteOrders();
         return(true);
        }
     }
   else
     {
      //--- 숏 포지션의 클로징 / 지정가 매도 주문 삭제 가능성을 확인합니다
      if(CheckCloseShort())
        {
         DeleteOrders();
         return(true);
        }
     }
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckCloseLong

롱 포지션을 클로즈 하기 위한 조건을 확인합니다.

```
virtual bool  CheckCloseLong()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

롱 포지션(시그널 객체의 CheckCloseLong() 메서드)를 클로즈하기 위한 조건을 확인하고 만족하면 오픈 포지션(CloseLong(...) 메서드)을 클로즈합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 롱 포지션 클로즈 또는 리밋/스탑 주문 삭제를 확인합니다          |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseLong()
  {
   double price=EMPTY_VALUE;
//--- 롱 클로즈 작업을 확인합니다
   if(m_signal.CheckCloseLong(price))
      return(CloseLong(price));
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckCloseShort

숏 포지션을 클로즈 하기 위한 조건을 확인합니다.

```
virtual bool  CheckCloseShort()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

숏 포지션(시그널 객체의 CheckCloseShort() 메서드)를 클로즈하기 위한 조건을 확인하고 만족하면 포지션(CloseShort() 메서드)를 클로즈 합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 숏 포지션 클로즈 또는 리밋/스탑 주문 삭제를 확인합니다          |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseShort()
  {
   double price=EMPTY_VALUE;
//--- 숏 클로즈 작업을 확인합니다
   if(m_signal.CheckCloseShort(price))
      return(CloseShort(price));
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseAll

전체 포지션 클로즈의 일부를 수행합니다.

```
virtual bool  CloseAll(
   double    lot    // lot
   )
```

Parameters

lot

[in] 포지션을 줄일 로트 수.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

"넷팅"모드에서는, CExpertTrade::Buy 또는 CExpertTrade::Sell 멧드를 사용하여 포지션을 클로즈합니다. "헤징" 모드에서는 CTrade::PositionClose 메서드가 사용되며, 이는 넷팅 모드를 사용하는 계정에서도 사용할 수 있습니다. DeleteOrders() 메서드는 주문 삭제에 사용됩니다.

Implementation

```
//+------------------------------------------------------------------+
//| 포지션 클로즈 및 주문 삭제                                       |
//| INPUT:  lot - 클로즈 볼륨.                                       |
//| OUTPUT: 거래 작업이 처리되면 true, 그렇지 않으면 false.          |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CloseAll(double lot)
  {
   bool result=false;
//--- 클로즈 작업을 확인합니다
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
     {
      if(m_position.PositionType()==POSITION_TYPE_BUY)
         result=m_trade.Sell(lot,0,0,0);
      else
         result=m_trade.Buy(lot,0,0,0);
     }
   result|=DeleteOrders();
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

Close

오픈 포지션을 클로즈 합니다.

```
virtual bool  Close()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

CTrade 클래스 객체의 포지션(PositionClose() 메서드)을 클로즈합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 포지션 클로즈                                                    |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseLong

Closes the long position.

```
virtual bool  CloseLong(
   double    price    // 가격
   )
```

Parameters

price

[in] 시장 진입가.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

"넷팅" 모드에서는 CExpertTrade::Buy 또는 CExpertTrade::Sell 메서드를 사용하여 포지션을 클로즈합니다. "헤징" 모드에서는 CTrade::PositionCloseByTicket 메서드가 사용됩니다.

Implementation

```
//+------------------------------------------------------------------+
//| 롱 포지션 클로즈                                                 |
//| INPUT:  price - 종가.                                            |
//| OUTPUT: 거래 작업이 처리되면 true, 그렇지 않으면 false.          |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   bool result=false;
//---
   if(price==EMPTY_VALUE)
      return(false);
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
      result=m_trade.Sell(m_position.Volume(),price,0,0);
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

CloseShort

숏 포지션을 클로즈합니다

```
virtual bool  CloseShort(
   double    price    // 가격
   )
```

Parameters

price

[in] 시장 진입가.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

"넷팅" 모드에서는 CExpertTrade::Buy 또는 CExpertTrade::Sell 메서드를 사용하여 포지션을 클로즈합니다. "헤징" 모드에서는 CTrade::PositionCloseByTicket 메서드가 사용됩니다.

Implementation

```
//+------------------------------------------------------------------+
//| 숏 포지션 클로즈                                                 |
//| INPUT:  price - 종가.                                            |
//| OUTPUT: 거래 작업이 성공적이면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   bool result=false;
//---
   if(price==EMPTY_VALUE)
      return(false);
   if(m_margin_mode==ACCOUNT_MARGIN_MODE_RETAIL_HEDGING)
      result=m_trade.PositionCloseByTicket(m_position.Identifier());
   else
      result=m_trade.Buy(m_position.Volume(),price,0,0);
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStop

이것은 오픈 포지션의 추적 손절매 조건을 확인합니다.

```
virtual bool  CheckTrailingStop()
```

반환 값

거래 작업이 실행되었으면 true, 아니면 false.

참고

오픈 포지션의 추적 손절매 조건(롱 또는 숏 포지션의 경우CheckTrailingStopLong() 또는 CheckTrailingStopShort())을 확인합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 추적 손절매/이익 실현 포지션을 확인합니다                                  |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.              |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStop()
  {
//--- 포지션은 호출 전에 선택되어야 합니다
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- 롱 포지션 수정 가능성을 확인합니다
      if(CheckTrailingStopLong()) return(true);
     }
   else
     {
      //--- 숏 포지션 수정 가능성을 확인합니다
      if(CheckTrailingStopShort()) return(true);
     }
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

오픈된 롱 포지션의 추적 손절매 조건을 확인합니다.

```
virtual bool  CheckTrailingStopLong()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

오픈된 롱 포지션의 추적 손절매 조건(Expert 추적 개체의 CheckTrailingStopLong(...) 메서드)을 확인합니다. 조건이 충족되면 포지션 파라미터(TrailingStopLong(...) 메서드)를 수정합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 추적 손절매/이익 실현 포지션을 확인합니다                        |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopLong()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- 롱 추적 손절매 작업을 확인합니다
   if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- 롱 추적 손절매 작업
      return(TrailingStopLong(sl,tp));
     }
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

오픈된 숏 포지션의 추적 손절매 조건을 확인합니다.

```
virtual bool  CheckTrailingStopShort()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

오픈된 숏 포지션의 추적 손절매 조건(Expert 추적 개체의 CheckTrailingStopShort(...) 메서드)을 확인합니다. 조건이 충족되면 포지션 파라미터(TrailingStopShort(...) 메서드)를 수정합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 추적 손절매/이익 실현 포지션을 확인합니다                        |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopShort()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- 숏 추적 손절매 작업을 확인합니다
   if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
     {
      if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
      if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
      //--- 추적 손절매 작업
      return(TrailingStopShort(sl,tp));
     }
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingStopLong

오픈된 롱 포지션의 파라미터를 수정합니다.

```
virtual bool  TrailingStopLong(
   double    sl,    // 스탑 로스
   double    tp,    // 이익 실현
   )
```

Parameters

sl

[in] 스탑 로스 가격.

tp

[in] 이익 실현 가격.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

이 함수는 오픈된 롱 포지션(CTrade 클래스 객체의 PositionModify(...) 메서드)의 파라미터를 수정합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 추적 손절매/이익 롱 포지션                                       |
//| INPUT:  sl - 새 스탑로스,                                        |
//|         tp - 새 이익실현.                                        |
//| OUTPUT: 거래 작업 성공시 true, 그렇지 않으면 false.              |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopLong(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingStopShort

오픈된 숏 포지션의 파라미터를 수정합니다.

```
virtual bool  TrailingStopLong(
   double    sl,    // 스탑 로스
   double    tp,    // 이익 실현
   )
```

Parameters

sl

[in] 스탑 로스 가격.

tp

[in] Take Profit price.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

함수는 오픈된 숏 포지션(CTrade 클래스 객체의 PositionModify(...) 메서드)의 파라미터를 수정합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 추적 손절매/이익 숏 포지션                                       |
//| INPUT:  sl - 새 스탑로스,                                        |
//|         tp - 새 이익실현.                                        |
//| OUTPUT: 거래 작업 성공시 true, 그렇지 않으면 false.              |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopShort(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderLong

리밋/스탑 매수 주문의 추적 손절매 조건을 확인합니다.

```
virtual bool  CheckTrailingOrderLong()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

추적 손절매 조건에서 리밋/스탑 추적 주문(거래 시그널 개체의 CheckTrailingOrderLong() 메서드)를 확인하고 필요한 경우 주문 파라미터를 수정합니다(TrailingOrderLong(...) 메서드).

Implementation

```
//+------------------------------------------------------------------+
//| 리밋/스탑 롱 오더 추적을 확인합니다                                |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.        |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderLong()
  {
   double price;
//--- 롱 주문 수정 가능성을 확인합니다
   if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
      return(TrailingOrderLong(m_order.PriceOpen()-price));
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderShort

리밋/스탑 매도 추적 주문의 추적 손절매 조건을 확인합니다.

```
virtual bool  CheckTrailingOrderShort()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

리밋/스탑 매도 추적 주문(거래 시그널 객체의 CheckTrailingOrderShort() 메서드)에 대한 추적 손절매 조건을 확인하고 필요한 경우 주문 파라미터를 수정합니다(TrailingOrderShort() 메서드).

Implementation

```
//+------------------------------------------------------------------+
//| 리밋/스탑 숏 추적 주문을 확인합니다                                |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.        |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderShort()
  {
   double price;
//--- 숏 주문 수정 가능성을 확인합니다
   if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
      return(TrailingOrderShort(m_order.PriceOpen()-price));
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingOrderLong

매수 제한/추적 손절매 주문의 매개변수를 수정합니다.

```
virtual bool  TrailingOrderLong(
   double    delta    // 델타
   )
```

Parameters

delta

[in] 가격 델타.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

매수 제한/추적 손절매 주문(CTrade 클래스 객체의 OrderModify(...) 메서드) 매개 변수를 수정합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 추적 롱 리밋/스탑 주문                                           |
//| INPUT:  delta - 가격 변동.                                       |
//| OUTPUT: 거래 작업 성공시 true, 그렇지 않으면 false.              |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderLong(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- 롱 주문 수정
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2025, MetaQuotes Ltd.

TrailingOrderShort

매도 제한/추적 손절매 주문의 파라미터를 수정합니다.

```
virtual bool  TrailingOrderShort(
   double    delta    // 델타
   )
```

Parameters

delta

[in] 가격 델타.

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

이것은 매도 제한/ 추적 주문(CTrade 클래스 객체의 OrderModify(...) 메서드)의 매개 변수를 수정합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 리밋/스탑 추적 숏 주문/stop order                                  |
//| INPUT:  delta - 가격 변동.                                       |
//| OUTPUT: 거래 작업 성공시 true, 그렇지 않으면 false.              |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::TrailingOrderShort(double delta)
  {
   ulong  ticket=m_order.Ticket();
   double price =m_order.PriceOpen()-delta;
   double sl    =m_order.StopLoss()-delta;
   double tp    =m_order.TakeProfit()-delta;
//--- 숏 주문 수정
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckDeleteOrderLong

리밋/스탑 매수 주문 조건을 확인합니다.

```
virtual bool  CheckDeleteOrderLong()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

주문 만료 시간을 확인합니다. 리밋/스탑 매수 주문(시그널 클래스 객체의 CheckCloseLong(...) 메서드) 삭제 조건을 확인하고 조건이 충족되면 주문(DeleteOrderLong() 메서드)을 삭제합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 롱 리밋/스탑 주문 삭제 확인                                      |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderLong()
  {
   double price;
//--- 롱 주문의 삭제 가능성을 확인합니다
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderLong());
     }
   if(m_signal.CheckCloseLong(price))
      return(DeleteOrderLong());
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

CheckDeleteOrderShort

리밋/스탑 매도 주문 삭제 조건을 확인합니다.

```
virtual bool  CheckDeleteOrderShort()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

1. 주문 만료 시간을 확인합니다.

2. 리밋/스탑 매도 주문(시그널 클래스 객체의 CheckCloseShort(...) 메서드) 삭제 조건을 확인하고 조건 중 하나가 충족되면 순서(DeleteOrderShort() 메서드)를 삭제합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 숏 리밋/스탑 주문 삭제 확인                                      |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 진행된다면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderShort()
  {
   double price;
//--- 숏 주문 삭제 가능성을 확인합니다
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderShort());
     }
   if(m_signal.CheckCloseShort(price))
      return(DeleteOrderShort());
//--- 작업 없이 반환합니다
   return(false);
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrders

모든 주문을 삭제합니다.

```
virtual bool  DeleteOrders()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

이는 모든 주문을 삭제합니다(모든 주문에 대한 DeleteOrder()).

Implementation

```
//+------------------------------------------------------------------+
//| 모든 리밋/스탑 주문 삭제                                         |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 성공적이면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrders()
  {
   bool result=false;
   int  total=OrdersTotal();
//---
   for(int i=total-1;i>=0;i--)
     {
      if(m_order.Select(OrderGetTicket(i)))
        {
         if(m_order.Symbol()!=m_symbol.Name()) continue;
         result|=DeleteOrder();
        }
     }
//---
   return(result);
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrder

리밋/스탑 주문을 삭제합니다.

```
virtual bool  DeleteOrder()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

리밋/스탑 주문(CTrade 클래스 개체의 OrderDelete(...) 메서드)를 삭제합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 리밋/스탑 주문 삭제                                              |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 성공적이면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrder()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrderLong

리밋/스탑 매수 주문을 삭제합니다.

```
virtual bool  DeleteOrderLong()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

리밋/스탑 매수 주문(CTrade 클래스 오브젝트의 OrderDelete(...) 메서드)를 삭제합니다.

Implementation

```
//+------------------------------------------------------------------+
//| 리밋/스탑 롱 주문을 삭제합니다                                   |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 성공적이면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderLong()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2025, MetaQuotes Ltd.

DeleteOrderShort

리밋/스탑 매도 주문을 삭제합니다.

```
virtual bool  DeleteOrderShort()
```

반환 값

거래 작업이 수행되었으면 true, 그게 아니면 false.

참고

리밋/스탑 매도 주문을 삭제합니다(CTrade 클래스 객체의 OrderDelete(...) 메서드).

Implementation

```
//+------------------------------------------------------------------+
//| 숏 리밋/스탑 주문 삭제                                           |
//| INPUT:  no.                                                      |
//| OUTPUT: 거래 작업이 성공적이면 true, 그렇지 않으면 false.       |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderShort()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotOpenLong

매수 작업 거래량을 가져옵니다.

```
double  LotOpenLong(
   double    price,   // 가격
   double    sl       // 스탑 로스
   )
```

Parameters

price

[in] 시장 진입가격.

sl

[in] 스탑 로스 가격.

반환 값

구매 작업을 위한 거래량(로트 단위).

참고

매수 작업(자금 관리 객체의 CheckOpenLong(...) 메서드) 거래량을 가져옵니다.

Implementation

```
//+------------------------------------------------------------------+
//| 오픈 롱 포지션에 대한 로트 획득 메서드.                          |
//| INPUT:  price - price,                                           |
//|         sl    - 손실 제한.                                       |
//| OUTPUT: 오픈용 로트.                                             |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
double CExpert::LotOpenLong(double price,double sl)
  {
   return(m_money.CheckOpenLong(price,sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotOpenShort

매도 작업 거래량을 가져옵니다.

```
double  LotOpenShort(
   double    price,   // 가격
   double    sl       // 스탑 로스
   )
```

Parameters

price

[in] 시장 진입가격.

sl

[in] 스탑 로스 가격.

반환 값

매도 작업용 거래량(로트 단위).

참고

매도 작업(자금 관리 객체의 CheckOpenShort(...)) 거래량을 가져옵니다.

Implementation

```
//+------------------------------------------------------------------+
//| 오픈 숏 포지션 로트 획득 방법.                                   |
//| INPUT:  price - 가격,                                            |
//|         sl    - 스탑로스.                                        |
//| OUTPUT: 오픈용 로트.                                             |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
double CExpert::LotOpenShort(double price,double sl)
  {
   return(m_money.CheckOpenShort(price,sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

LotReverse

포지션 리버스용 거래량을 가져옵니다.

```
double  LotReverse(
   double    sl       // 스탑 로스
   )
```

Parameters

sl

[in] 스탑 로스 가격.

반환 값

포지션 리버스 작업용 거래량(로트 단위).

참고

포지션 리버스 작업(자금 관리 객체의 CheckReverse(...) 메서드)용 거래량을 가져옵니다.

Implementation

```
//+------------------------------------------------------------------+
//| 리버스 포지션용 로트를 가져오는 메서드.                          |
//| INPUT:  sl - 스탑 로스.                                          |
//| OUTPUT: 오픈용 로트.                                             |
//| 비고: no.                                                        |
//+------------------------------------------------------------------+
double CExpert::LotReverse(double sl)
  {
   return(m_money.CheckReverse(GetPointer(m_position),sl));
  }
```

© 2000-2025, MetaQuotes Ltd.

PrepareHistoryDate

거래 내역 추적을 위한 시작 날짜를 설정합니다.

```
void  PrepareHistoryDate()
```

참고

거래 내역 추적 기간은 월 초부터 설정됩니다(단, 하루 이상).

© 2000-2025, MetaQuotes Ltd.

HistoryPoint

거래 내역의 체크포인트를 생성합니다.

```
void  HistoryPoint(
   bool   from_check_trade=false   // 플래그
   )
```

Parameters

from_check_trade=false
[in] 재귀 방지 플래그.

참고

이는 포지션, 주문, 거래 및 과거 주문의 양을 저장합니다.

© 2000-2025, MetaQuotes Ltd.

CheckTradeState

현재 상태를 저장된 상태와 비교하고 해당 이벤트 핸들러를 호출합니다.

```
bool  CheckTradeState()
```

반환 값

이벤트가 처리되면 true, 그렇지 않으면 false.

참고

HistoryPoint() 메서드로 저장한 값과 비교하여 포지션, 주문, 거래 및 과거 주문 수를 확인합니다. 거래 기록이 변경된 경우 해당 가상 이벤트 핸들러를 호출합니다.

© 2000-2025, MetaQuotes Ltd.

WaitEvent

이벤트 대기 플래그를 설정합니다.

```
void  WaitEvent(
   ENUM_TRADE_EVENTS    event          // 플래그
   )
```

Parameters

event

[in] 설정할 이벤트를 기다리는 플래그(ENUM_TRADE_EVENTS 열거형으로부터).

반환 값

None.

이벤트 플래그

```
//--- 예상되는 이벤트의 플래그
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT                =0,         // 예상되는 이벤트 없음
   TRADE_EVENT_POSITION_OPEN           =0x1,       // "포지션 오픈"이 기대되는 플래그
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,         // "포지션 볼륨 수정" 이벤트가 기대되는 플
   TRADE_EVENT_POSITION_MODIFY         =0x4,       // "포지션 스탑 주문 수정" 이벤트가 기대되
   TRADE_EVENT_POSITION_CLOSE          =0x8,       // "포지션 클로징" 이벤트가 기대되는 플래
   TRADE_EVENT_POSITION_STOP_TAKE      =0x10,      // "포지션 스탑 주문 촉발"이 기대되는 플래
   TRADE_EVENT_ORDER_PLACE             =0x20,      // "지정가 주문 실행" 이벤트가 기대되는 플
   TRADE_EVENT_ORDER_MODIFY            =0x40,      // "지정가 주문 수정" 이벤트가 기대되는 플
   TRADE_EVENT_ORDER_DELETE            =0x80,      // "지정가 주문 삭제" 이벤트가 기대되는 플
   TRADE_EVENT_ORDER_TRIGGER           =0x100      // "지정가 주문 촉발" 이벤트가 기대되는 플
  };
```

© 2000-2025, MetaQuotes Ltd.

NoWaitEvent

이벤트 대기 플래그를 재설정합니다.

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS    event          // 플래그
   )
```

Parameters

event

[in] 재설정할 이벤트(ENUM_TRADE_EVENTS 열거)로 플래그 지정.

반환 값

None.

이벤트 플래그

```
//--- 예상되는 이벤트의 플래그
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT                =0,         // 예상되는 이벤트 없음
   TRADE_EVENT_POSITION_OPEN           =0x1,       // "포지션 오픈"이 기대되는 플래그
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,         // "포지션 볼륨 수정" 이벤트가 기대되는 플
   TRADE_EVENT_POSITION_MODIFY         =0x4,       // "포지션 스탑 주문 수정" 이벤트가 기대되
   TRADE_EVENT_POSITION_CLOSE          =0x8,       // "포지션 클로징" 이벤트가 기대되는 플래
   TRADE_EVENT_POSITION_STOP_TAKE      =0x10,      // "포지션 스탑 주문 촉발"이 기대되는 플래
   TRADE_EVENT_ORDER_PLACE             =0x20,      // "지정가 주문 실행" 이벤트가 기대되는 플
   TRADE_EVENT_ORDER_MODIFY            =0x40,      // "지정가 주문 수정" 이벤트가 기대되는 플
   TRADE_EVENT_ORDER_DELETE            =0x80,      // "지정가 주문 삭제" 이벤트가 기대되는 플
   TRADE_EVENT_ORDER_TRIGGER           =0x100      // "지정가 주문 촉발" 이벤트가 기대되는 플
  };
```

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionStopTake

"포지션 스탑 로스/이익 실현 트리거" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventPositionStopTake()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderTriggered

"지정가 주문 트리거" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventOrderTriggered()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionOpened

"포지션 오픈" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventPositionOpened()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionVolumeChanged

"포지션 볼륨 변경" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventPositionVolumeChanged()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionModified

"포지션 수정" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventPositionModified()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventPositionClosed

"클로즈 포지션" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventPositionClosed()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderPlaced

"지정가 주문 실행" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventOrderPlaced()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderModified

"지정가 주문 수정" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventOrderModified()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventOrderDeleted

"지정가 주문 삭제" 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventOrderDeleted()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

TradeEventNotIdentified

식별되지 않은 이벤트의 이벤트 핸들러.

```
virtual bool  TradeEventNotIdentified()
```

반환 값

CExpert 클래스 메서드는 아무 작업도 수행하지 않으며 항상 true를 반환합니다.

참고

몇 가지 거래 이벤트가 발생할 수 있으며, 이러한 경우에는 이를 식별하기가 어렵습니다.

© 2000-2025, MetaQuotes Ltd.

TimeframeAdd

추적용 타임프레임을 추가합니다.

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period          // 타임프레임
   )
```

Parameters

period

[in] 추적할 타임프레임(ENUM_TIMEFRAMES 열거값으로부터).

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

TimeframesFlags

타임프레임 플래그를 구성합니다.

```
int  TimeframesFlags(
   MqlDateTime&    time          // 레퍼런스
   )
```

Parameters

time

[in] 새 시간을 포함하는 MqlDateTime 유형 구조에 대한 레퍼런스.

반환 값

새 막대가 있는 시간대를 나타내는 플래그를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CExpertSignal

CExpertSignal은 트레이딩 신호의 기본 클래스이며 (CheckReverseLong() 및 CheckReverseShort() 메서드를 제외하고) 아무 작업도 하지 않고 인터페이스를 제공합니다.

사용 방법:

1. 시그널을 교환하기 위한 알고리즘을 준비합니다;
2. CExpertSignal 클래스에서 상속된 고유한 거래 시그널 클래스를 만듭니다;
3. 클래스의 가상 메서드를 사용자 알고리즘으로 재정의합니다.

ExpertWSignalW폴더에서 시그널 클래스의 예를 찾을 수 있습니다.

Description

CExpertSignal은 거래 시그널 알고리즘 구현을 위한 기본 클래스입니다.

Declaration

```
class CExpertSignal : public CExpertBase
```

Title

```
#include <Expert\ExpertSignal.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertSignal

직계 하위

CSignalAC, CSignalAMA, CSignalAO, CSignalBearsPower, CSignalBullsPower, CSignalCCI, CSignalDeM, CSignalDEMA, CSignalEnvelopes, CSignalFrAMA, CSignalRSI, CSignalRVI, CSignalSAR, CSignalStoch, CSignalTEMA, CSignalTriX, CSignalWPR

그룹별 클래스 메서드

Initialization	
virtual InitIndicators	지표 및 시간대 초기화
virtual ValidationSettings	개체 설정을 확인합니다
virtual AddFilter	결합된 시그널에 필터를 추가합니다
보호 데이터로 액세스	
BasePrice	기본 가격 수준을 설정합니다
UsedSeries	사용된 시계열 플래그를 가져옵니다
파라미터 설정	

© 2000-2025, MetaQuotes Ltd.

Initialization	
Weight	"Weight" 파라미터 값을 설정합니다
PatternsUsage	"PatternsUsage" 파라미터 값을 설정합니다
General	"General" 파라미터 값을 설정합니다
Ignore	"Ignore" 파라미터 값을 설정합니다
Invert	"Invert" 파라미터 값을 설정합니다
ThresholdOpen	"ThresholdOpen" 파라미터 값을 설정합니다
ThresholdClose	"ThresholdClose" 파라미터 값을 설정합니다
PriceLevel	"PriceLevel" 파라미터 값을 설정합니다
StopLevel	"StopLevel" 파라미터 값을 설정합니다
TakeLevel	"TakeLevel" 파라미터 값을 설정합니다
Expiration	"Expiration" 파라미터 값을 설정합니다
Magic	"Magic" 파라미터 값을 설정합니다
거래 조건 확인	
virtual CheckOpenLong	롱 포지션 오픈 상태를 점검합니다
virtual CheckCloseLong	롱 포지션을 클로즈하기 위한 조건을 점검합니다
virtual CheckOpenShort	숏 포지션 오픈 조건을 점검합니다
virtual CheckCloseShort	숏 포지션을 클로즈하기 위한 조건을 점검합니다
virtual CheckReverseLong	롱 포지션 리버스의 조건을 확인합니다
virtual CheckReverseShort	숏 포지션 리버스의 조건을 확인합니다
Trade Parameters Setting	
virtual OpenLongParams	롱 포지션 오프닝을 위한 파라미터를 설정합니다
virtual OpenShortParams	숏 포지션 오프닝을 위한 파라미터를 설정합니다
virtual CloseLongParams	롱 포지션 클로징을 위한 파라미터를 설정합니다
virtual CloseShortParams	숏 포지션 클로징을 위한 파라미터를 설정합니다
주문 추적 조건을 확인합니다	
virtual CheckTrailingOrderLong	지정가 매수 주문의 매개 변수를 수정하기 위한 조건을 확인합니다
virtual CheckTrailingOrderShort	지정가 매도 주문의 매개 변수를 수정하기 위한 조건을 확인합니다
시장가 주문 형성을 확인하는 방법	

© 2000-2025, MetaQuotes Ltd.

Initialization	
virtual LongCondition	매수 조건을 확인한 결과를 가져옵니다
virtual ShortCondition	매도 조건을 확인한 결과를 가져옵니다.
virtual Direction	가격에 대한 "가중" 디렉션을 가져옵니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

BasePrice

기준 가격 수준을 설정합니다.

```
void  BasePrice(
   double    value          // 값
   )
```

Parameters

value

[in] 파라미터의 새 값.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

UsedSeries

사용된 시간대의 플래그를 가져옵니다.

```
int  UsedSeries()
```

반환 값

사용된 시계열의 플래그(심볼/타임프레임이 작업 심볼/타임프레임에 해당하는 경우), 그렇지 않으면 0입니다.

© 2000-2025, MetaQuotes Ltd.

Weight

"Weight" 매개 변수의 새 값을 설정합니다.

```
void  Weight(
   double    value         // 값
   )
```

Parameters

value

[in] "Weight" 파라미터.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

PatternUsage

"PatternsUsage" 파라미터의 새 값을 설정합니다.

```
void  PatternUsage(
   double    value         // 값
   )
```

Parameters

value

[in] "PatternsUsage"의 새 값.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

General

"General" 매개 변수의 새 값을 설정합니다.

```
void  General(
   int     value          // 값
   )
```

Parameters

value

[in] "General 새 값.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

Ignore

"Ignore" 파라미터의 새 값을 설정합니다.

```
void  Ignore(
   long     value          // 값
   )
```

Parameters

value

[in] "Ignore"의 새 값.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

Invert

"Invert" 매개 변수의 새 값을 설정합니다.

```
void  Invert(
   long     value          // 값
   )
```

Parameters

value

[in] "Invert"의 새 값.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

ThresholdOpen

"ThresholdOpen" 매개 변수의 새 값을 설정합니다.

```
void  ThresholdOpen(
   long     value          // 값
   )
```

Parameters

value

[in] "ThresholdOpen"의 새 값.

반환 값

None.

참고

"ThresholdOpen" 매개 변수의 범위는 0에서 100까지입니다. 오픈 포지션으로 "투표"할 때 사용합니다.

© 2000-2025, MetaQuotes Ltd.

ThresholdClose

"ThresholdClose" 매개 변수의 새 값을 설정합니다.

```
void  ThresholdOpen(
   long     value          // 값
   )
```

Parameters

value

[in] "ThresholdClose"의 새 값.

반환 값

None.

참고

ThresholdClose 매개 변수의 범위는 0부터 100까지입니다. 클로즈 포지션으로 "투표"할 때 사용합니다.

© 2000-2025, MetaQuotes Ltd.

PriceLevel

"PriceLevel" 파라미터의 새 값을 설정합니다.

```
void  PriceLevel(
   double    value          // 값
   )
```

Parameters

value

[in] "PriceLevel"의 새 값.

반환 값

None.

참고

"PriceLevel"의 값은 가격 수준 단위로 정의됩니다. "PriceLevel"의 값은 가격 수준 단위로 정의됩니다. "PriceLevel"은 기준 가격에 대한 시가를 정의하는 데 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

StopLevel

"StopLevel" 파라미터의 새 값을 설정합니다.

```
void  StopLevel(
   double    value          // 값
   )
```

Parameters

value

[in] "StopLevel"의 새 값.

반환 값

None.

참고

"StopLevel"의 값은 가격 수준 단위로 정의됩니다. "PriceLevel"의 값은 가격 수준 단위로 정의됩니다. "StopLevel"은 시가에 대한 스탑 로스 가격을 정의하는 데 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

TakeLevel

"TakeLevel" 매개 변수의 새 값을 설정합니다.

```
void  TakeLevel(
   double    value          // 값
   )
```

Parameters

value

[in] "TakeLevel"의 새 값.

반환 값

None.

참고

"TakeLevel"의 값은 가격 수준 단위로 정의됩니다. "PriceLevel"의 값은 가격 수준 단위로 정의됩니다. "TakeLevel"은 시가에 비례하여 이익 창출 가격을 정의하는 데 사용됩니다

© 2000-2025, MetaQuotes Ltd.

Expiration

"만료" 매개 변수의 값을 설정합니다.

```
void  Expiration(
   int     value          // 값
   )
```

Parameters

value

[in] "만료"의 새 값.

반환 값

None.

참고

"만료" 매개 변수의 값은 막대로 정의됩니다. 지정가 주문의 만료 시간(지정가 주문을 사용하여 거래할 때)으로 사용됩니다.

© 2000-2025, MetaQuotes Ltd.

Magic

"Magic" 파라미터의 값을 설정합니다.

```
void  Magic(
   int     value          // 값
   )
```

Parameters

value

[in] "Magic"의 새 값 (Expert Advisor ID).

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

객체 설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

객체 설정이 올바르면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

필요한 모든 지표 및 시계열을 초기화합니다.

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터.

반환 값

성공적으로 완료되면 true, 그렇지 않으면 false.

참고

개체가 초기화할 때 정의된 것과 다른 심볼 또는 시계열을 사용하는 경우에만 필요한 타임프레임이 초기화됩니다.

© 2000-2025, MetaQuotes Ltd.

AddFilter

복합 신호에 필터를 추가합니다.

```
virtual bool  AddFilter(
   CExpertSignal*  filter    // 포인터
   )
```

Parameters

indicators

[in] 필터 객체에 대한 포인터.

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

롱 포지션을 오픈하기 위한 조건을 확인합니다.

```
virtual bool  CheckOpenLong(
   double&    price,          // 가격
   double&    sl,             // 스탑 로스
   double&    tp,             // 이익 실현
   datetime&  expiration      // 만료
   )
```

Parameters

price

[in][out] 레퍼런스로 전달된 시가에 대한 변수.

sl

[in][out] 레퍼런스로 전달된 스탑 로스에 대한 변수.

tp

[in][out] 레퍼런스로 전달된 이익 실현 가격에 대한 변수.

expiration

[in][out] 레퍼런스로 전달된 만료 시간에 대한 변수 (필요시).

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

숏 포지션을 오픈하기 위한 조건을 확인합니다.

```
virtual bool  CheckOpenShort(
   double&    price,          // 가격
   double&    sl,             // 스탑 로스
   double&    tp,             // 이익 실현
   datetime&  expiration      // 만료
   )
```

Parameters

price

[in][out] 레퍼런스로 전달된 시가에 대한 변수.

sl

[in][out] 레퍼런스로 전달된 스탑 로스에 대한 변수.

tp

[in][out] 레퍼런스로 전달된 이익 실현 가격에 대한 변수.

expiration

[in][out] 레퍼런스로 전달된 만료 시간에 대한 변수 (필요시).

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

OpenLongParams

롱 포지션을 오픈하도록 파라미터를 설정합니다.

```
virtual bool  OpenLongParams(
   double&    price,          // 가격
   double&    sl,             // 스탑 로스
   double&    tp,             // 이익 실현
   datetime&  expiration      // 만료
   )
```

Parameters

price

[in][out] 레퍼런스로 전달된 시가에 대한 변수.

sl

[in][out] 레퍼런스로 전달된 스탑 로스에 대한 변수.

tp

[in][out] 레퍼런스로 전달된 이익 실현 가격에 대한 변수.

expiration

[in][out] 레퍼런스로 전달된 만료 시간에 대한 변수 (필요시).

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

OpenShortParams

숏 포지션을 오픈하도록 파라미터를 설정합니다.

```
virtual bool  OpenShortParams(
   double&    price,          // 가격
   double&    sl,             // 스탑 로스
   double&    tp,             // 이익 실현
   datetime&  expiration      // 만료
   )
```

Parameters

price

[in][out] 레퍼런스에 의해 전달된 가격을 위한 변수.

sl

[in][out] 레퍼런스로 전달된 스탑 로스에 대한 변수.

tp

[in][out] 레퍼런스로 전달된 이익 실현 가격에 대한 변수.

expiration

[in][out] 레퍼런스로 전달된 만료 시간에 대한 변수 (필요시).

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckCloseLong

롱 포지션을 클로즈 하기 위한 조건을 확인합니다.

```
virtual bool  CheckCloseLong(
   double&  price      // 가격
   )
```

Parameters

price

[in][out] 레퍼런스로 전달된 종가의 변수.

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckCloseShort

숏 포지션을 클로즈 하기 위한 조건을 확인합니다.

```
virtual bool  CheckCloseShort(
   double&  price      // 가격
   )
```

Parameters

price

[in][out] 레퍼런스로 전달된 종가의 변수.

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CloseLongParams

롱 포지션을 클로즈 하기 위한 파라미터를 설정합니다.

```
virtual bool  CloseLongParams(
   double&    price          // 가격
   )
```

Parameters

price

[in][out] 레퍼런스로 전달된 종가의 변수.

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CloseShortParams

숏 포지션을 클로즈 하기 위한 파라미터를 설정합니다.

```
virtual bool  CloseShortParams(
   double&    price          // 가격
   )
```

Parameters

price

[in][out] 레퍼런스로 전달된 종가의 변수.

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckReverseLong

롱 포지션 리버스 조건을 확인합니다.

```
virtual bool  CheckReverseLong(
   double&    price,          // 가격
   double&    sl,             // 스탑 로스
   double&    tp,             // 이익 실현
   datetime&  expiration      // 만료
   )
```

Parameters

price

[in][out] 레퍼런스에 의해 전달된 가격을 위한 변수.

sl

[in][out] 레퍼런스로 전달된 스탑 로스에 대한 변수.

tp

[in][out] 레퍼런스로 전달된 이익 실현 가격에 대한 변수.

expiration

[in][out] 레퍼런스로 전달된 만료 시간에 대한 변수 (필요시).

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckReverseShort

숏 포지션 리버스의 조건을 확인합니다.

```
virtual bool  CheckReverseShort(
   double&    price,          // 가격
   double&    sl,             // 스탑 로스
   double&    tp,             // 이익 실현
   datetime&  expiration      // 만료
   )
```

Parameters

price

[in][out] 레퍼런스로 전달된 리버스 가격을 위한 변수.

sl

[in][out] 레퍼런스로 전달된 스탑 로스에 대한 변수.

tp

[in][out] 레퍼런스로 전달된 이익 실현 가격에 대한 변수.

expiration

[in][out] 레퍼런스로 전달된 만료 시간에 대한 변수 (필요시).

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderLong

지정가 매수 주문의 매개 변수를 수정하기 위한 조건을 확인합니다.

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // 주문
   double&         price           // 가격
   )
```

Parameters

order

[in] COrderInfo 클래스 객체에 대한 포인터.

price

[in][out] 스탑 로스 가격에 대한 변수.

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingOrderShort

지정가 매도 주문의 매개 변수를 수정하기 위한 조건을 확인합니다.

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // 주문
   double&         price           // 가격
   )
```

Parameters

order

[in] COrderInfo 클래스 객체에 대한 포인터.

price

[in][out] 스탑 로스 가격에 대한 변수.

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

LongCondition

롱 포지션을 오픈하기 위한 조건을 확인합니다.

```
virtual int  LongCondition()
```

반환 값

조건이 충족되면 (시그널 "강도"에 따라) 1에서 100 사이의 값을 반환합니다. 롱 포지션을 오픈 하는 시그널이 없으면 0을 반환합니다.

참고

기본 클래스는 롱 포지션을 오픈하는 확인 조건을 구현하지 않으며 항상 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

ShortCondition

숏 포지션을 오픈하기 위한 조건을 확인합니다.

```
virtual int  ShortCondition()
```

반환 값

조건이 충족되면 (시그널 "강도"에 따라) 1에서 100 사이의 값을 반환합니다. 숏 포지션을 오픈하는 시그널이 없으면 0을 반환합니다.

참고

기본 클래스는 숏 포지션을 오픈하는 확인 조건을 구현하지 않으며 항상 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Direction

"가중된" 가격의 디렉션 값을 반환합니다.

```
virtual double  Direction()
```

반환 값

상향 방향이 가장 유력할 경우 값 >0을 반환하고 하향 방향일 경우 <0을 반환합니다. 절대값은 시그널의 "강도"에 따라 달라집니다.

참고

내장 필터를 사용하는 경우 일반 방향을 정의할 때 결과가 고려됩니다.

© 2000-2025, MetaQuotes Ltd.

CExpertTrailing

CExpertTrailing은 추적 알고리즘의 기본 클래스이며 인터페이스를 제공할 뿐입니다.

사용 방법:

1. 추적 알고리즘 준비;
2. CExpertTrailing 클래스에서 상속된 자신만의 추적 클래스 만들기;
3. 클래스의 가상 메서드를 사용자 알고리즘으로 재정의합니다.

여러분은 Expert₩railing₩에서 추적 클래스의 예를 찾을 수 있습니다.

Description

CExpertTrailing은 추적 손절매 알고리즘의 구현을 위한 기본 클래스입니다.

Declaration

```
class CExpertTrailing : public CExpertBase
```

Title

```
#include <Expert\ExpertTrailing.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertTrailing

직계 하위

CTrailingFixedPips, CTrailingMA, CTrailingNone, CTrailingPSAR

그룹별 클래스 메서드

추적 손절매 상태 점검	
virtual CheckTrailingStopLong	롱 포지션의 파라미터를 수정하기 위한 조건을 점검합니다
virtual CheckTrailingStopShort	숏 포지션의 파라미터를 수정하기 위한 조건을 점검합니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

롱 포지션 파라미터를 수정하기 위한 조건을 확인합니다.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 스탑 로스
   double&         tp            // 이익 실현
   )
```

Parameters

position

[in] CPositionInfo 클래스 오브젝트.

sl

[in][out] 레퍼런스로 전달된 스탑 로스에 대한 변수.

tp

[in][out] 레퍼런스로 전달된 이익 실현 가격에 대한 변수.

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

참고

기본 클래스 메서드는 항상 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

롱 포지션의 파라미터를 수정하기 위한 조건을 확인합니다.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 스탑 로스
   double&         tp            // 이익 실현
   )
```

Parameters

position

[in] CPositionInfo 클래스 오브젝트.

sl

[in][out] 레퍼런스로 전달된 스탑 로스에 대한 변수.

tp

[in][out] 레퍼런스로 전달된 이익 실현 가격에 대한 변수.

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

참고

기본 클래스 메서드는 항상 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CExpertMoney

CExpertMoney는 비용 및 리스크 관리 알고리즘의 기본 클래스입니다.

Description

CExpertMoney는 자금 및 리스크 관리 클래스의 구현을 위한 기본 클래스입니다.

Declaration

```
class CExpertMoney : public CObject
```

Title

```
#include <Expert\ExpertMoney.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertMoney

직계 하위

CMoneyFixedLot, CMoneyFixedMargin, CMoneyFixedRisk, CMoneyNone, CMoneySizeOptimized

그룹별 클래스 메서드

보호 데이터로 액세스	
Percent	"위험률" 매개 변수의 값을 설정합니다.
Initialization	
virtual ValidationSettings	설정 확인
거래 조건 확인	
virtual CheckOpenLong	롱 포지션의 거래량을 가져옵니다
virtual CheckOpenShort	숏 포지션의 거래량을 가져옵니다
virtual CheckReverse	포지션 리버스 거래량을 가져옵니다
virtual CheckClose	오픈된 포지션을 클로즈 하기 위한 조건을 확인합니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

Percent

"위험률" 매개 변수의 값을 설정합니다.

```
void  Percent(
   double  percent      // 위험률
   )
```

Parameters

percent

[in] 위험률.

반환 값

None.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

성공이면 true, 그렇지 않으면 false.

참고

기본 클래스 메서드는 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

롱 포지션의 거래량을 가져옵니다.

```
virtual double  CheckOpenLong(
   double  price,     // 가격
   double  sl         // 스탑 로스
   )
```

Parameters

price

[in] 롱 포지션의 시가.

sl

[in] 롱 포지션의 스탑 로스 가격.

반환 값

롱 포지션의 거래량.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

숏 포지션의 거래량을 가져옵니다.

```
virtual double  CheckOpenShort(
   double  price,     // 가격
   double  sl         // 스탑 로스
   )
```

Parameters

price

[in] 숏 포지션을 위한 시작 가격.

sl

[in] 숏 포지션의 스탑 로스 가격.

반환 값

숏 포지션용 거래량.

© 2000-2025, MetaQuotes Ltd.

CheckReverse

포지션의 리버스 볼륨을 가져옵니다.

```
virtual double  CheckReverse(
   CPositionInfo*  position,     // 포인터
   double          sl            // 스탑 로스
   )
```

Parameters

position

[in] CPositionInfo 클래스 객체에 대한 포인터.

sl

[in] 스탑 로스 가격.

반환 값

포지션 리버스 볼륨.

© 2000-2025, MetaQuotes Ltd.

CheckClose

오픈된 포지션을 클로즈 하기 위한 조건을 확인합니다.

```
virtual double  CheckClose()
```

반환 값

조건이 만족되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

거래 시그널 모듈

클라이언트 터미널의 표준 제공에는 "MQL5 Wizard"에 대한 일련의 이미 만들어진 거래 시그널 모듈이 포함됩니다. MQL5 Wizard에서 Expert Advisor를 생성할 때 트레이드 신호 모듈(최대 64개)을 원하는 조합으로 사용할 수 있습니다. 거래 운영에 대한 최종 결정은 포함된 모든 모듈로부터 얻은 시그널의 복잡한 분석에 기초하여 이루어집니다. 무역 결정 메커니즘에 대한 자세한 설명은 아래에 제시되어 있습니다.

표준 전달에는 다음과 같은 시그널 모듈이 포함됩니다.

- 지표 가속기 오실레이터 시그널
- 적응 이동 평균 지표 시그널
- 어썸 오실레이터 지표 시그널
- 오실레이터 베어스 파워 시그널
- 오실레이터 불스 파워 시그널
- 오실레이터 상용 채널 지수 시그널
- 오실레이터 DeMarker 시그널
- 이중 지수 이동 평균 지표 시그널
- Envelopes 지표 시그널
- 프랙탈 적응형 이동평균 지표 시그널
- 하루 중 시간 필터 시그널
- Oscillator MACD 시그널
- 이동 평균 지표 시그널
- 파라볼릭 SAR 지표 시그널
- 오실레이터 상대적 강도 지수 시그널
- 오실레이터 상대적 활력 지수 시그널
- 확률적 오실레이터 시그널
- 오실레이터 삼중 지수 평균 시그널
- 삼중 지수 이동 평균 지표 시그널
- 오실레이터 윌리엄스 백분율 범위 시그널

시그널 모듈에 기초한 거래 의사결정 메커니즘

거래 의사결정 메커니즘은 다음과 같은 기본 원칙 목록으로 나타낼 수 있습니다:

- 각 시그널 모듈에는 시장 모듈 집합(가격과 지표 값의 확실한 조합)이 있습니다.
- 각 시장 모델은 1에서 100까지의 범위에 따라 달라질 수 있는 유의성을 가집니다. 유의성이 클수록 모형의 강도가 높아집니다.
- 각 모델은 가격 이동 방향에 대한 예측을 생성합니다.
- 모듈의 예측은 임베디드 모델을 검색한 결과이며 -100에서 100 사이의 숫자로 출력됩니다. 기호는 예측 이동 방향을 결정합니다(부정 기호는 가격이 하락할 것이라는 의미, 플러스 기호는 가격이 상승할 것이라는 의미). 절대값은 가장 잘 발견된 모형의 강도에 해당합니다.
- 각 모듈의 예측값은 설정("가중치")에 지정된 가중치 계수가 0-1인 최종 "투표"로 전송됩니다.

© 2000-2025, MetaQuotes Ltd.

- 투표 결과는 -100 ~ 100 범위 내의 숫자이며, 여기서 부호는 예측 움직임의 방향을 결정합니다. 이 값은 모든 시그널 모듈에 대한 가중 예측의 산술 평균으로 계산됩니다. 절대값은 Expert Advisor가 거래 결정을 내리는 데 사용됩니다.

생성된 각 Expert Advisor에는 0에서 100 사이의 값과 같을 수 있는 포지션(ThresholdOpen 및 ThresholdClose)의 오프닝 및 클로징 임계값 레벨의 두 가지 조정 가능한 설정이 있습니다. 최종 시그널의 강도가 임계값 수준을 초과하면 시그널의 부호에 해당하는 트레이드 연산이 수행됩니다.

예제

다음과 같은 임계값 수준을 가진 Expert Advisor를 염두해 보십시오: ThresholdOpen=20 및 ThresholdClose=90. 두 개의 신호 모듈이 무역 운영에 대한 결정에 참여하는데, 가중치가 0.4인 MA 모듈과 가중치가 0.8인 Stochastic 모듈입니다. 획득된 거래 신호의 두 가지 변수를 분석해봅시다:

Variant 1

가격이 상승하는 MA를 넘어섰습니다. 이 사례는 MA module에서 구현된 시장 모델 중 하나에 해당합니다. 이 모델은 가격 상승을 암시합니다. 그것의 중요성은 100과 같습니다. 동시에, 확률적 오실레이터는 기각했고 가격에 따라 차이를 형성했습니다. 이 사례는 확률적 모듈에 구현된 모델 중 하나에 해당합니다. 이 모델은 가격 하락을 암시합니다. 이 모델의 가중치는 80입니다.

최종 "투표"의 결과를 계산해 봅시다. MA 모듈에서 얻은 요율은 0.4 * 100 = 40으로 계산됩니다. 확률 모듈의 값은 0.8 *(-80) = -64로 계산됩니다. 최종 값은 (40 -64)/2 = -12의 두 비율의 산술 평균으로 계산됩니다. 투표 결과는 12와 같은 상대적인 강도로 매도 시그널입니다. 임계값 레벨(20)에 도달하지 못했습니다. 따라서 트레이드 작업이 수행되지 않습니다.

Variant 2

그 가격은 상승하는 MA를 넘어 하락세로 돌아섰습니다. 이 경우는 MA module에 구현된 모델 중 하나에 해당합니다. 이 모델은 가격 상승을 암시합니다. 그것의 중요성은 10과 같습니다. 동시에, 확률적 오실레이터는 기각했고 가격에 따라 차이를 형성했습니다. 이 사례는 확률적 모듈에 구현된 모델 중 하나에 해당합니다. 이 모델은 가격 하락을 암시합니다. 이 모델의 가중치는 80입니다.

최종 "투표"의 결과를 계산해 봅시다. MA 모듈에서 구한 요율은 0.4 * 10 = 4로 계산됩니다. 확률 모듈의 값은 0.8 *(-80) = -64로 계산됩니다. 최종 값은 (4 -64)/2 = -30의 두 비율의 산술 평균으로 계산됩니다. 투표 결과는 30과 같은 상대적인 강도로 매도하는 시그널입니다. 임계값 레벨(20)에 도달했습니다. 따라서 결과는 숏 포지션을 오픈하는 시그널입니다.

© 2000-2025, MetaQuotes Ltd.

a) 가격과 확률적 진동자(변수 1 및 2)의 차이.

b) 가격이 MA 지표를 초과(변수 1).

c) 가격이 MA 지표보다 하향(변수 2).

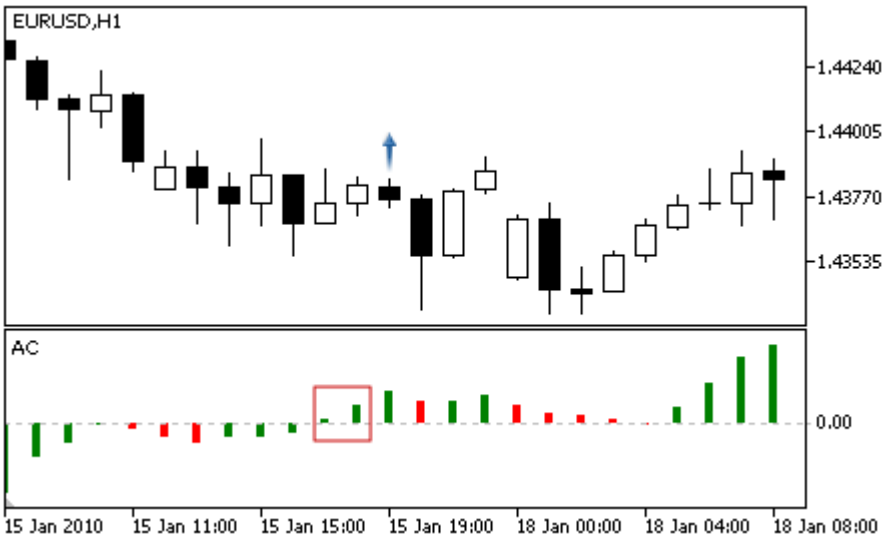
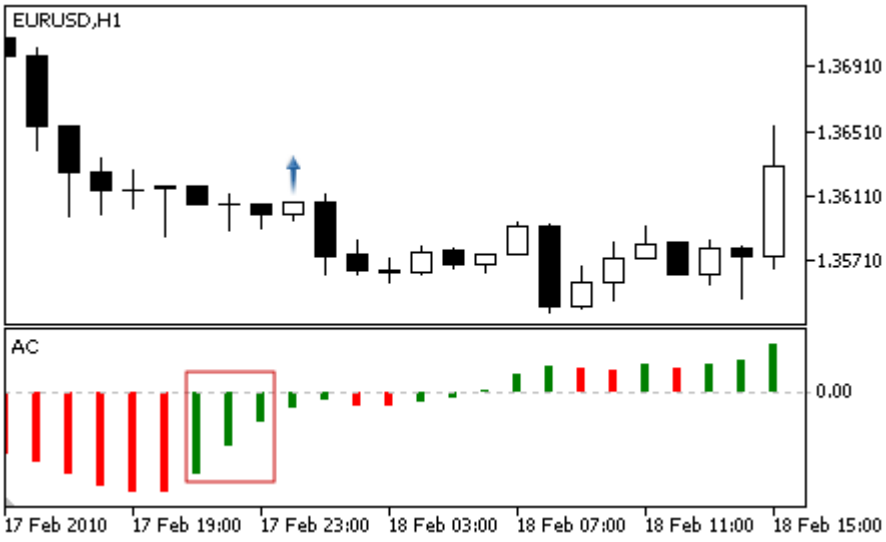
© 2000-2025, MetaQuotes Ltd.

지표 가속기 오실레이터 시그널

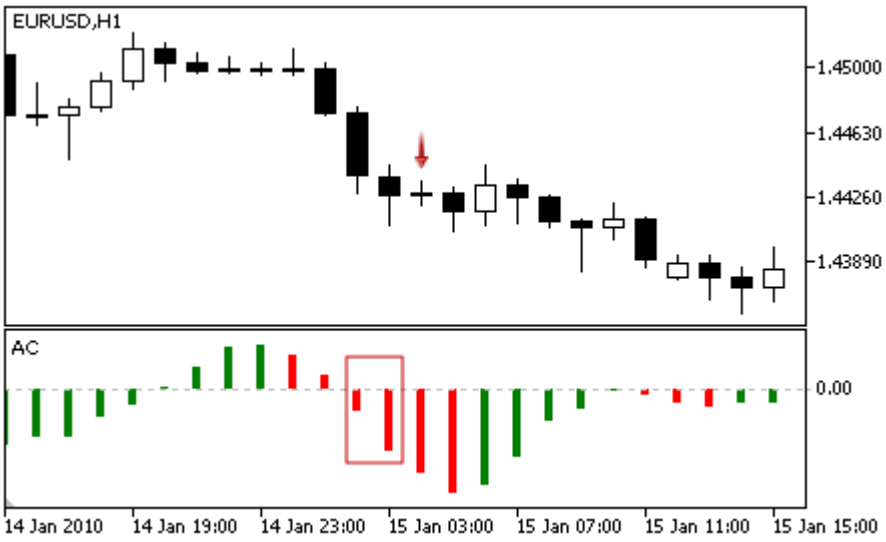
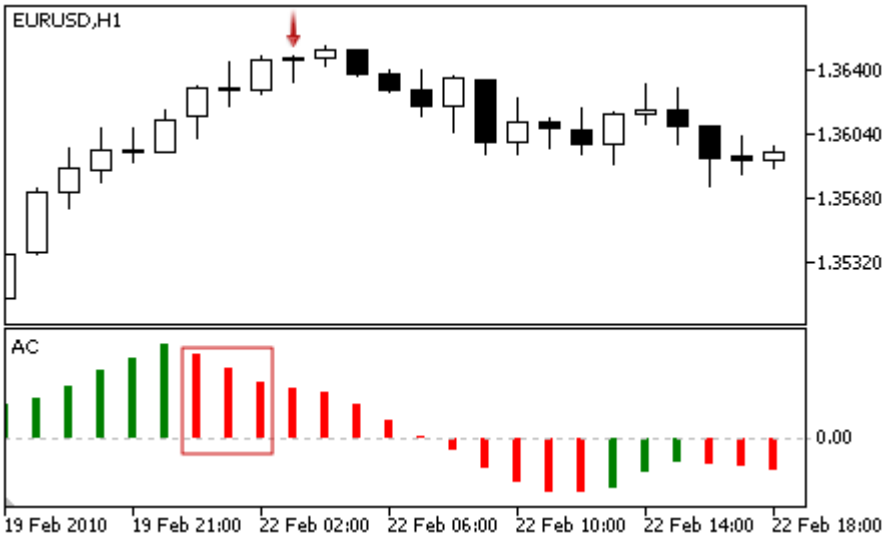
이 모듈은 가속기 오실레이터 지표의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• 지표 값은 0보다 크며 분석된 막대와 이전 막대에서 상승합니다.  • 지표 값은 0보다 작으며 분석된 막대와 이전 막대에서 상승합니다. 
매도용	• 지표 값은 0 이하이며 분석된 막대와 이전 막대에 해당됩니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 지표 값은 0보다 크며 분석된 두 개의 이전 막대에 해당됩니다. 
매수하는 데에 이의 없음	지표 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	지표 값은 막대에서 감소합니다.

참고

Expert Advisor의 작동 모드("모든 틱" 또는 "시가 전용")에 따라 분석 막대는 현재 막대(인덱스 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.

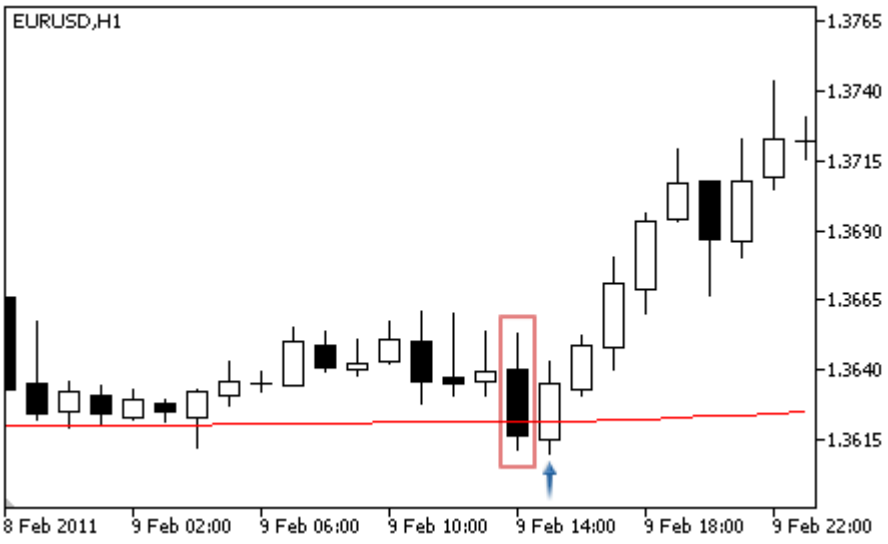
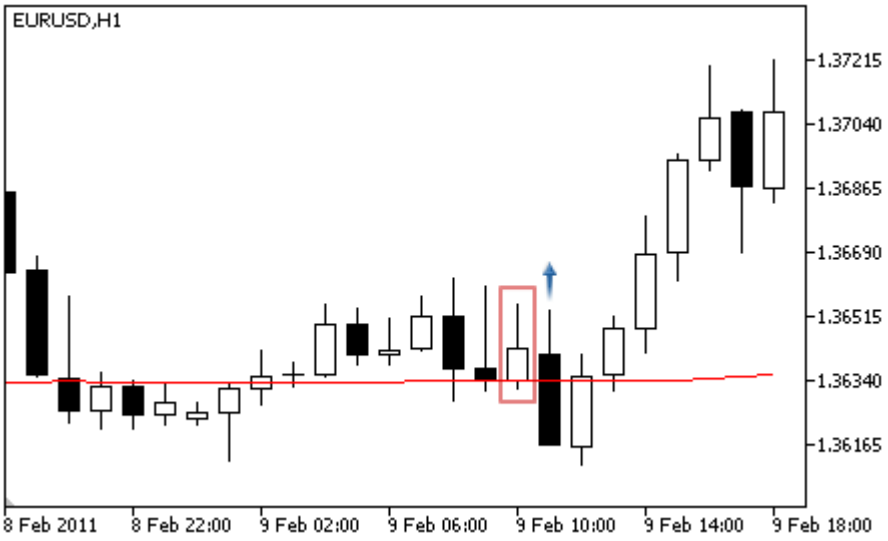
© 2000-2025, MetaQuotes Ltd.

적응 이동 평균 지표 시그널

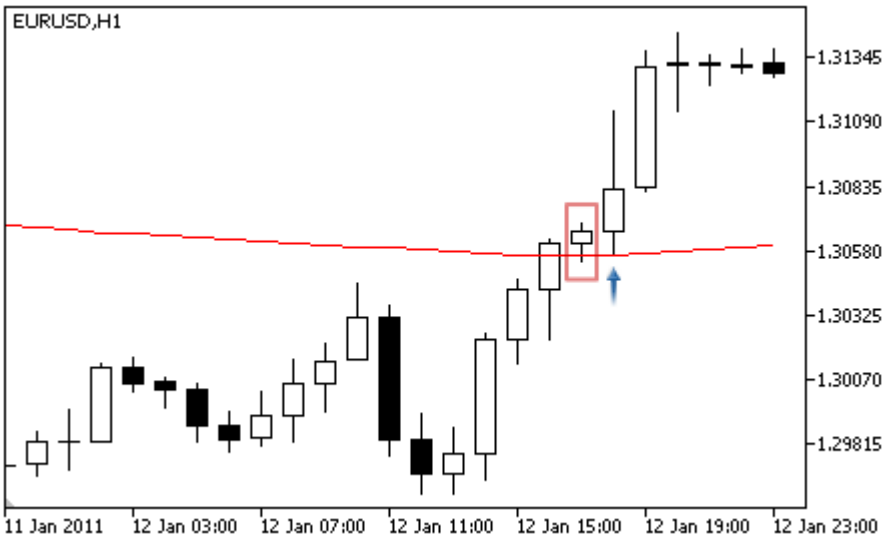
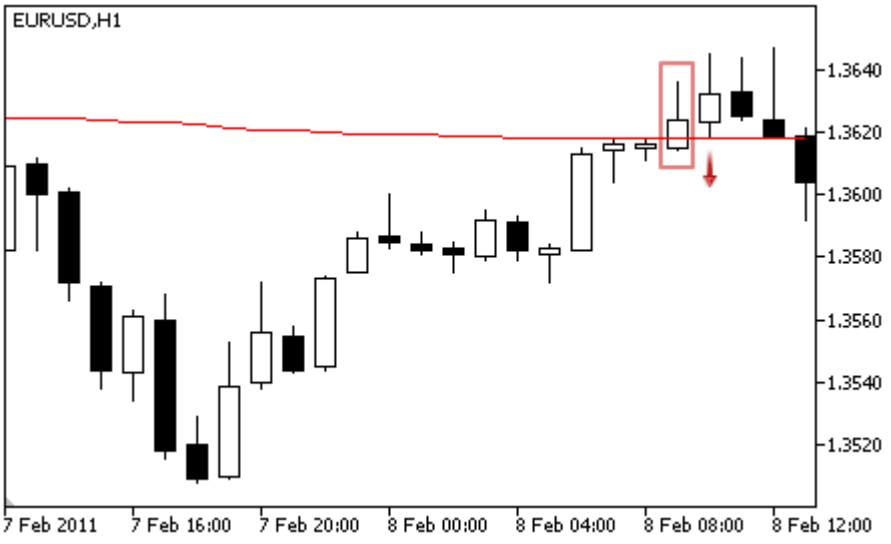
이 모듈은 적응 이동 평균 지표의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

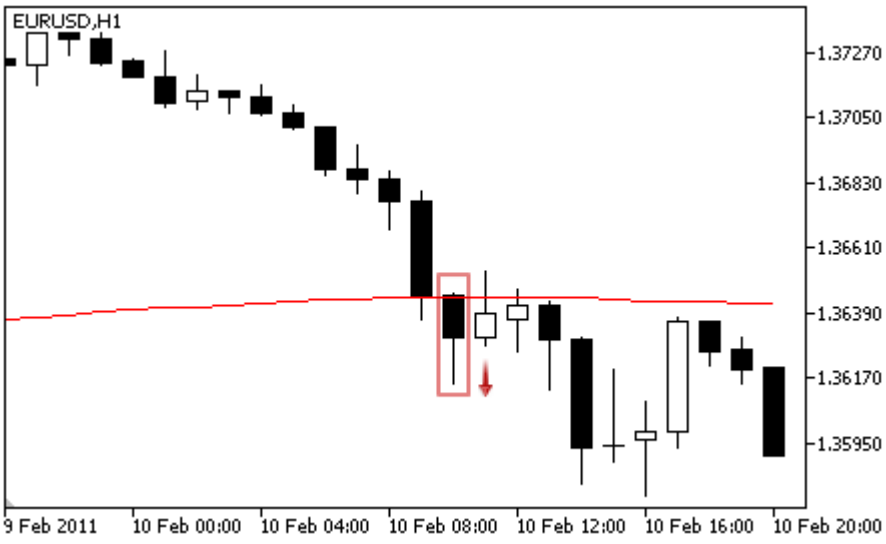
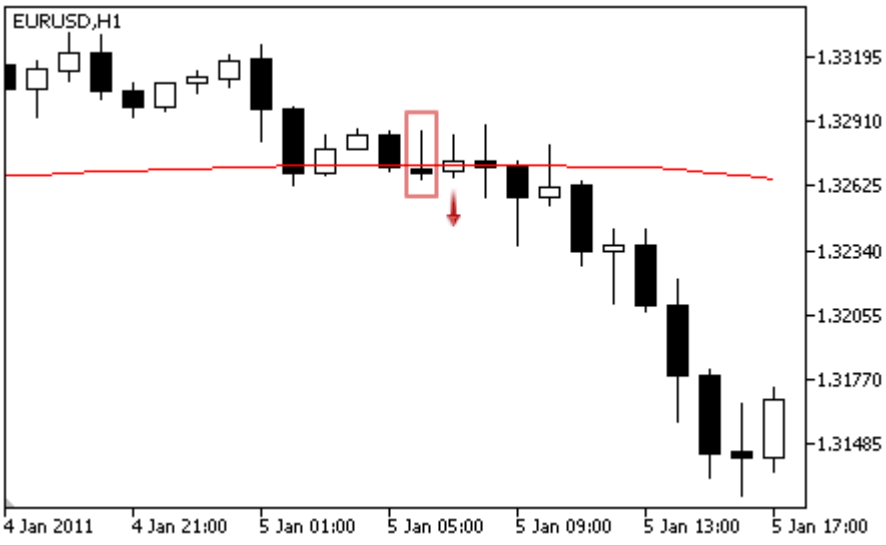
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• Failed 브레이크아웃. 가격이 지표 아래로 통과했지만(분석된 막대의 시가는 지표 위에 있고 종가는 지표 아래에 있음) 지표는 상승합니다(약세 지표 라인 롤백 신호).  • 이동 평균 교차. 가격이 지표를 위로 통과했습니다(분석된 막대의 시가는 지표보다 낮고 종가는 지표보다 높습니다). 지표가 상승합니다(강세 시그널). 

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	• True 브레이크아웃. 가격이 지표 위로 통과했지만(분석된 막대의 시가는 지표 아래에 있고 종가는 지표 위에 있음) 지표는 떨어집니다(약세 지표 라인 롤백 시그널). 
매도용	• Failed 브레이크아웃. 가격이 지표 위로 통과했지만(분석된 막대의 시가가 지표 아래에 있고 종가는 지표 위에 있음) 지표는 떨어집니다(약세 지표 라인 롤백 신호).  • 이동 평균 교차. 가격이 지표를 하향(분석된 막대의 시가는 지표보다 높고 종가는 지표보다 낮음)하고 지표는 하락(강세 시그널)합니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • True 브레이크아웃. 막대의 위쪽 그림자가 지표(분석된 막대의 시가 및 종가는 지표 아래에 있고, 고가는 지표 위에 있음)을 통과하고 지표가 떨어집니다(지표 선 롤백 신호). 
매수하는 데에 이의 없음	가격이 지표보다 위에 있습니다.
매도하는 데에 이의 없음	가격이 지표보다 아래에 있습니다.

참고

Expert Advisor의 작동 모드("모든 틱" 또는 "시가 전용")에 따라 분석 막대는 현재 막대(인덱스 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodMA	지표의 평균화 기간.
Shift	시간 축을 따라 지표를 이동합니다(막대).
Method	평균화 메서드.
Applied	지표 계산에 사용되는 가격 시리즈.

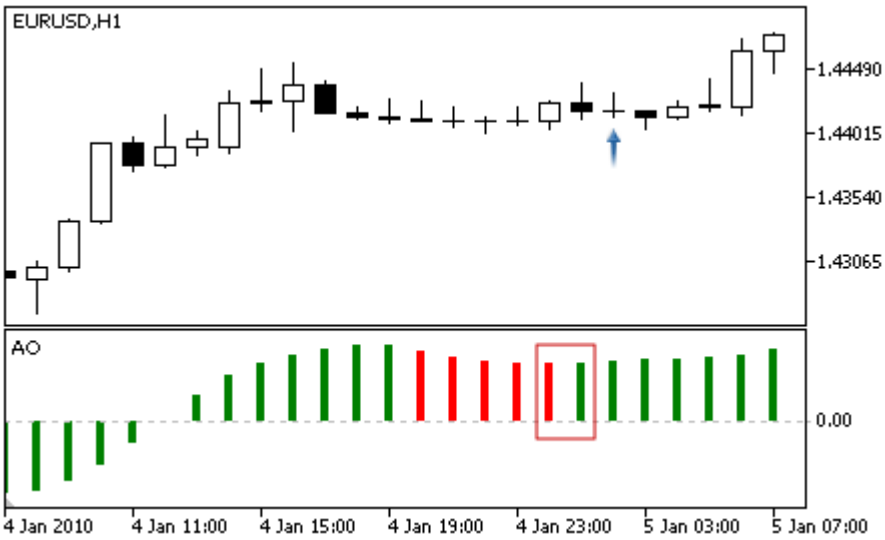
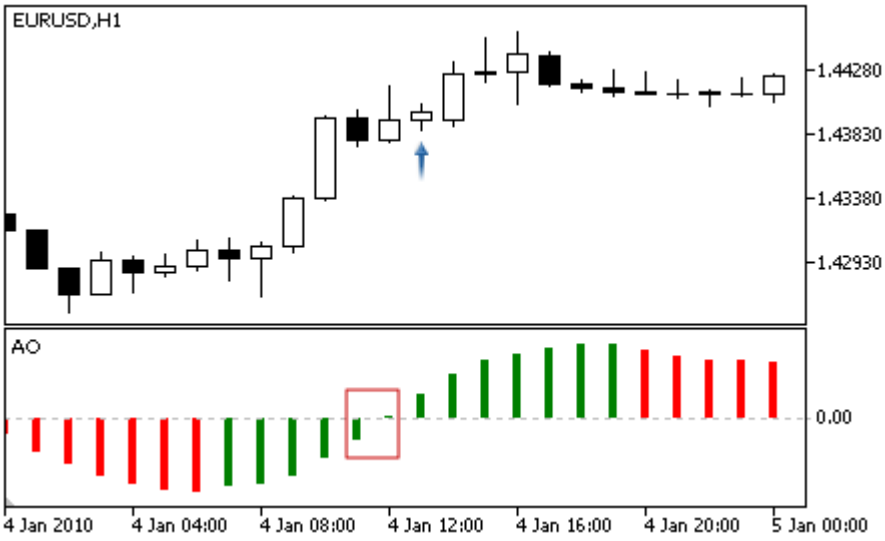
© 2000-2025, MetaQuotes Ltd.

어썸 오실레이터 지표 시그널

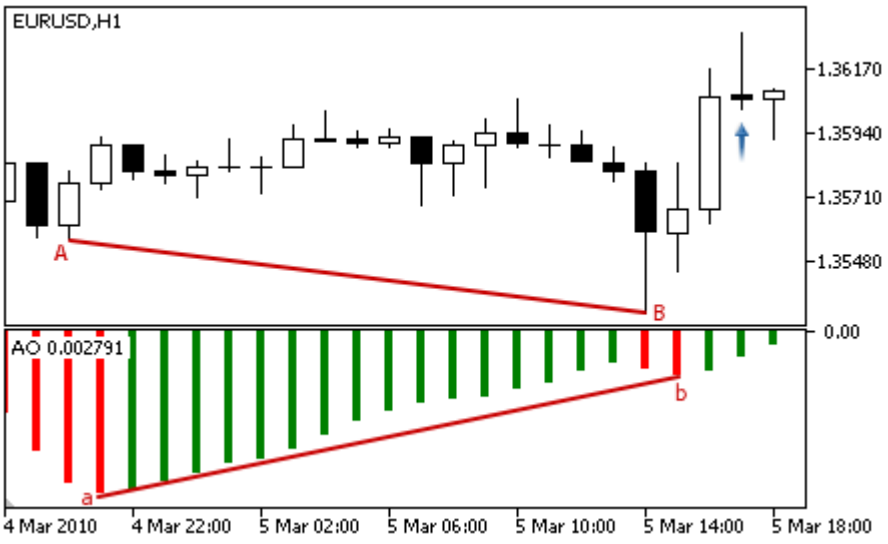
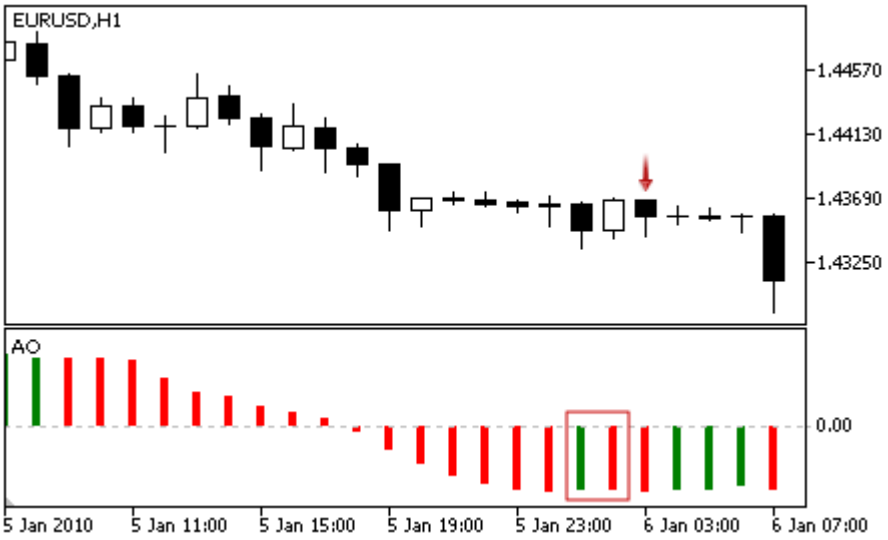
이 시그널 모듈은 지표 어썸 오실레이터시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

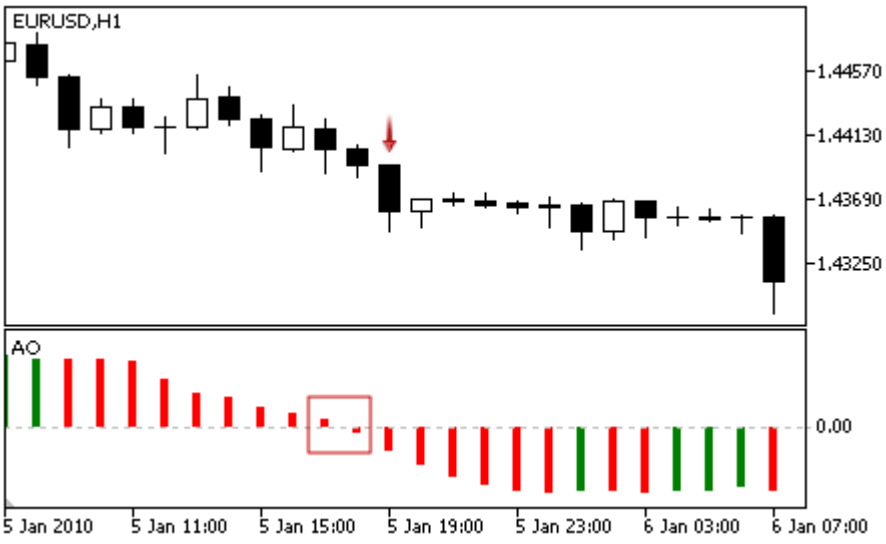
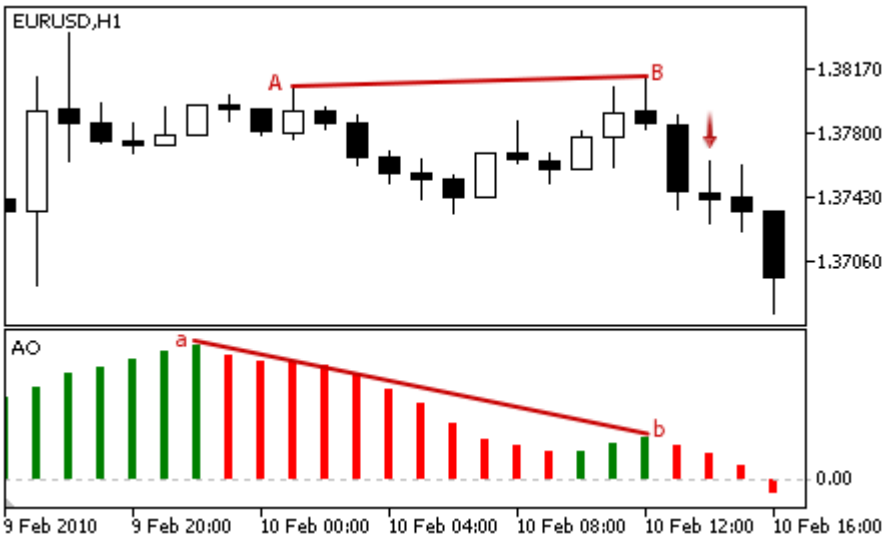
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **Saucer** —분석 막대의 지표 값이 상승하고 이전 막대의 지표 값이 떨어지는데, 이 경우 두 값 모두 0을 초과합니다.  • **제로 라인 교차** —지표 값은 분석 바에서 0 이상이고, 이전 바에서는 0 미만입니다.  • **Divergence** —첫 번째 분석된 지표의 바닥은 이전 지표보다 더 얕고, 해당 가격 바닥은 이전 지표보다 더 깊습니다. 또한 지표가 0 수준 이상으로 상승해서는 안 됩니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• **Saucer** —분석된 막대의 지표 값이 떨어지고 이전 막대의 지표 값이 상승하는데, 이 경우 두 값 모두 0보다 작습니다.  • **제로 라인 교차** —지표 값은 분석 바에서 0보다 낮고, 이전 바에서는 0보다 높습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • **Divergence** —지표의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다. 또한 지표가 영점 이하로 떨어져서는 안 됩니다. 
매수하는 데에 이의 없음	지표 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	지표 값은 막대에서 감소합니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.

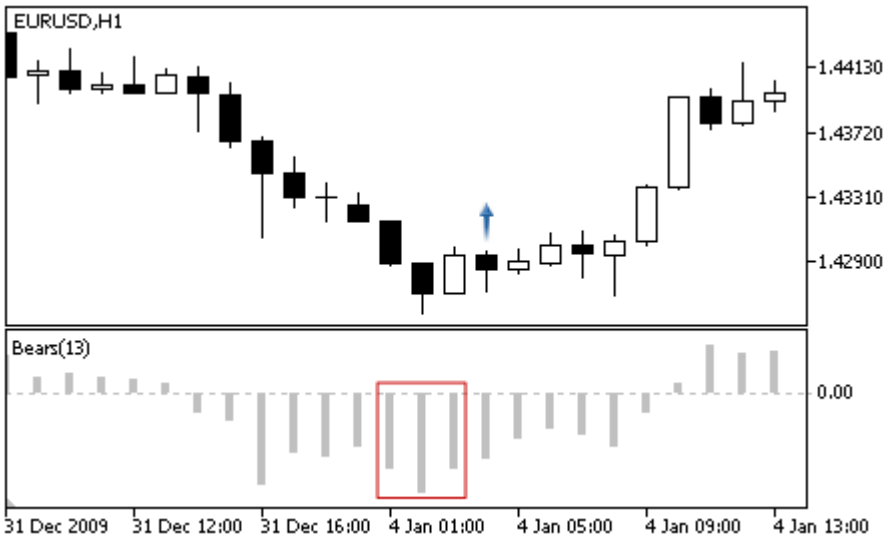
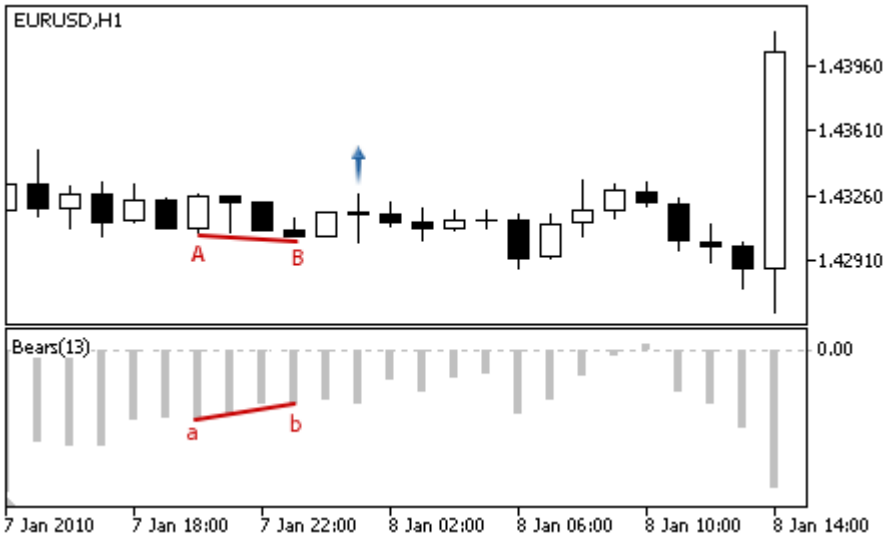
© 2000-2025, MetaQuotes Ltd.

오실레이터 베어스 파워 시그널

이 시그널 모듈은 오실레이터 베어스 파워의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• 리버스 — 오실레이터가 위쪽으로 회전하고 분석된 바의 값이 0 미만입니다.  • 확산 — 첫 번째 분석된 오실레이터의 바닥은 이전 것보다 높고 해당 가격 바닥은 이전 것보다 낮습니다. 또한 오실레이터가 0 레벨 이상으로 상승해서는 안 됩니다. 
매도용	매도 시그널 없음.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
매수하는 데에 이의 없음	오실레이터 값이 0보다 작습니다.
매도하는 데에 이의 없음	시그널 없음.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodBears	오실레이터의 계산 주기.

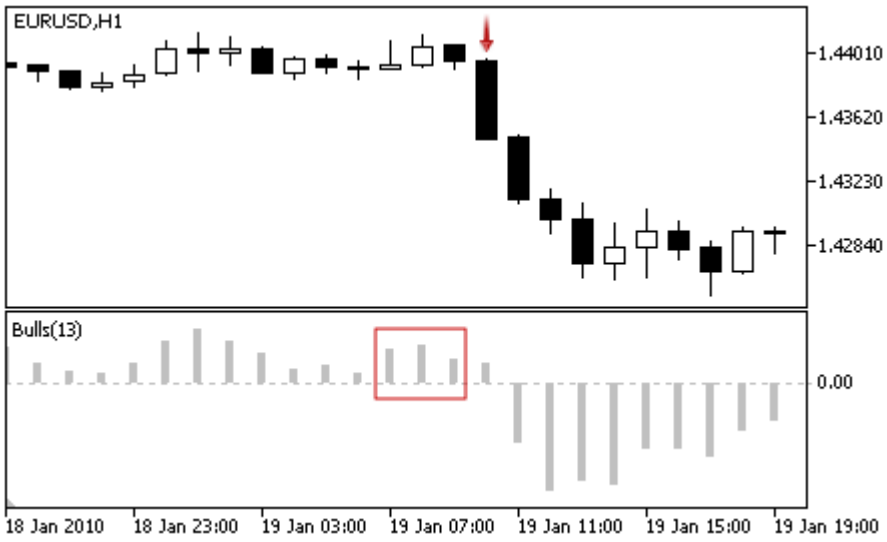
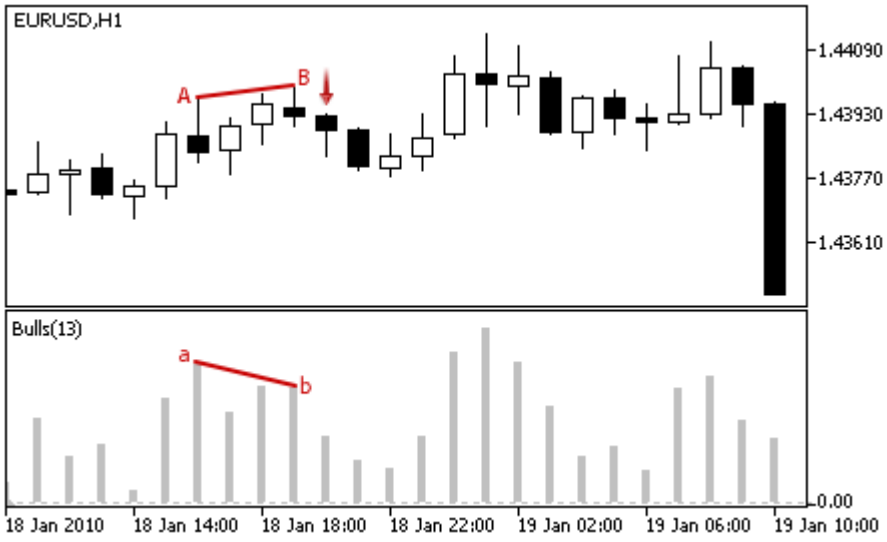
© 2000-2025, MetaQuotes Ltd.

오실레이터 불스 파워 시그널

이 시그널 모듈은 오실레이터 불스 파워의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	매도 시그널 없음.
매도용	• 리버스 — 오실레이터가 아래로 향했고 분석된 바의 값이 0보다 높습니다.  • 확산 — 오실레이터의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다. 또한 오실레이터가 0 레벨 아래로 내려가지 않아야 합니다. 
매수하는 데에 이의 없음	시그널 없음.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
매도하는 데에 이의 없음	오실레이터 값이 0보다 큽니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodBulls	오실레이터의 계산 주기.

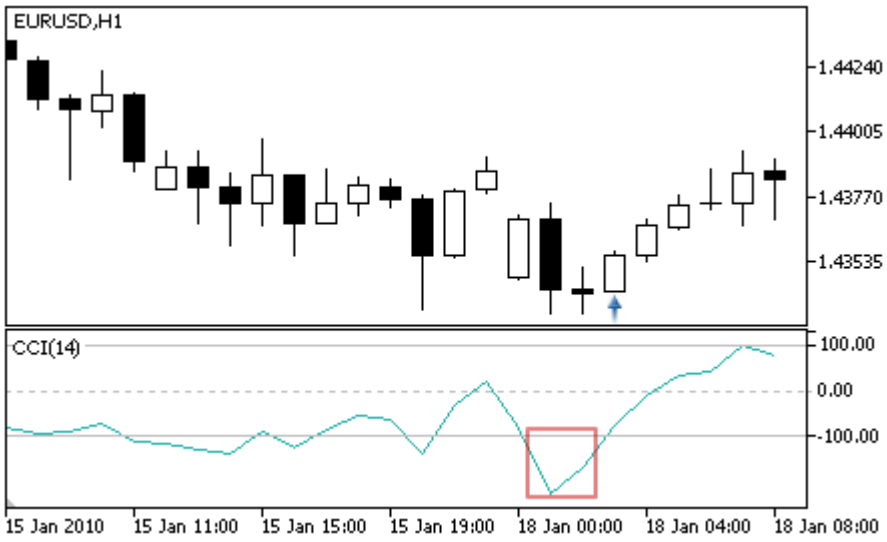
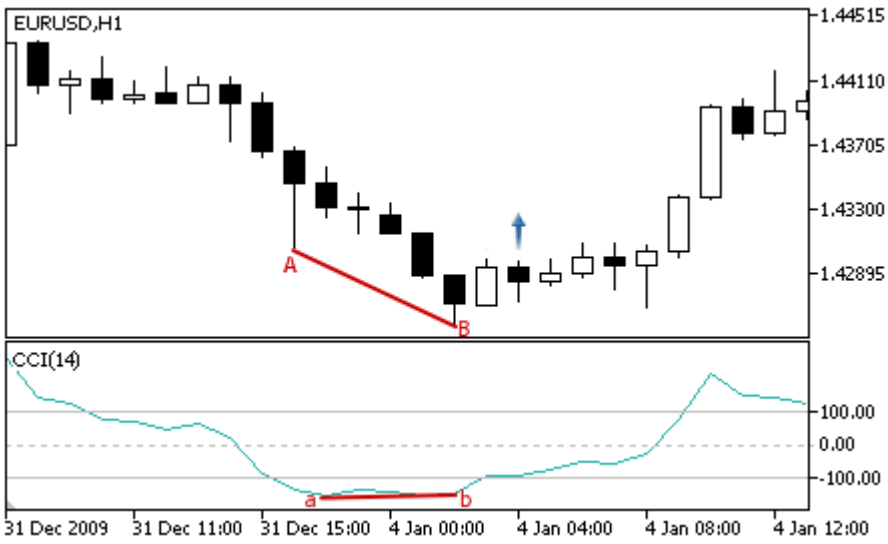
© 2000-2025, MetaQuotes Ltd.

오실레이터 상용 채널 지수 시그널

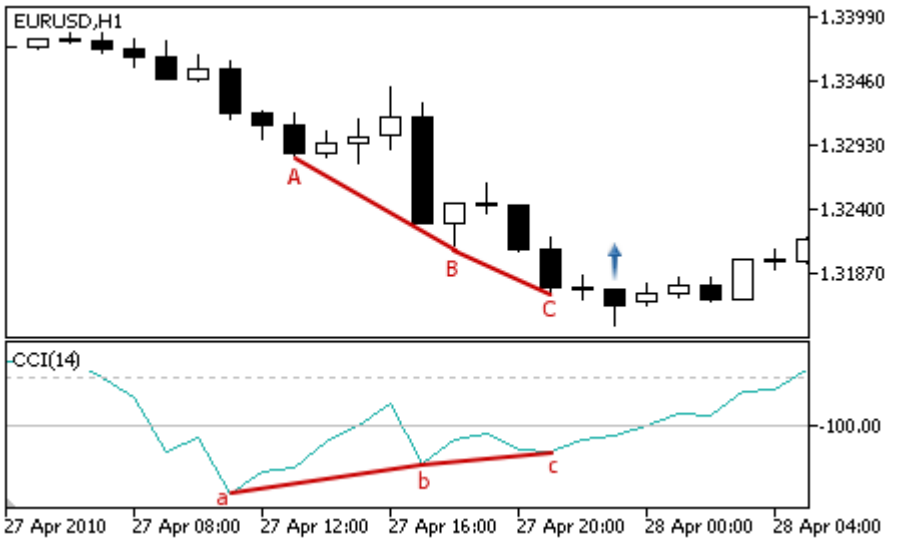
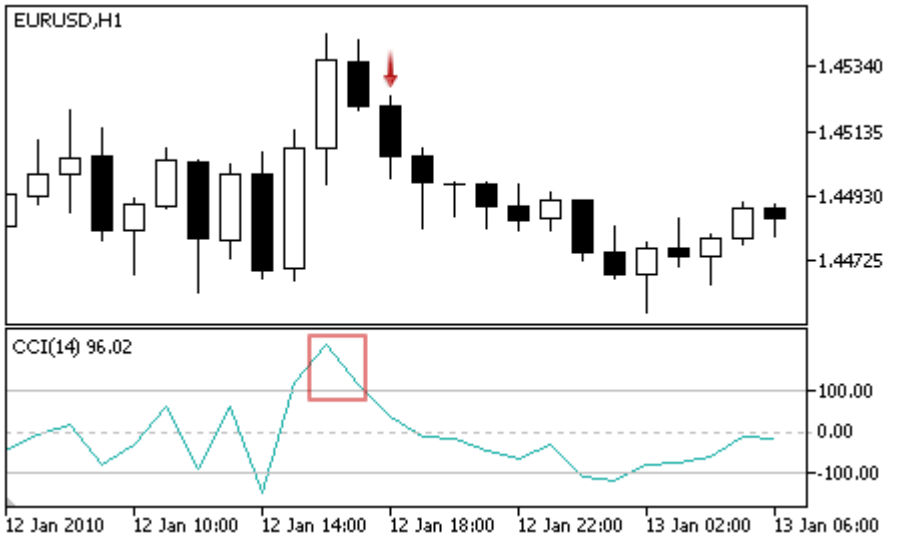
이 시그널 모듈은 오실레이터 상용 채널 지수 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

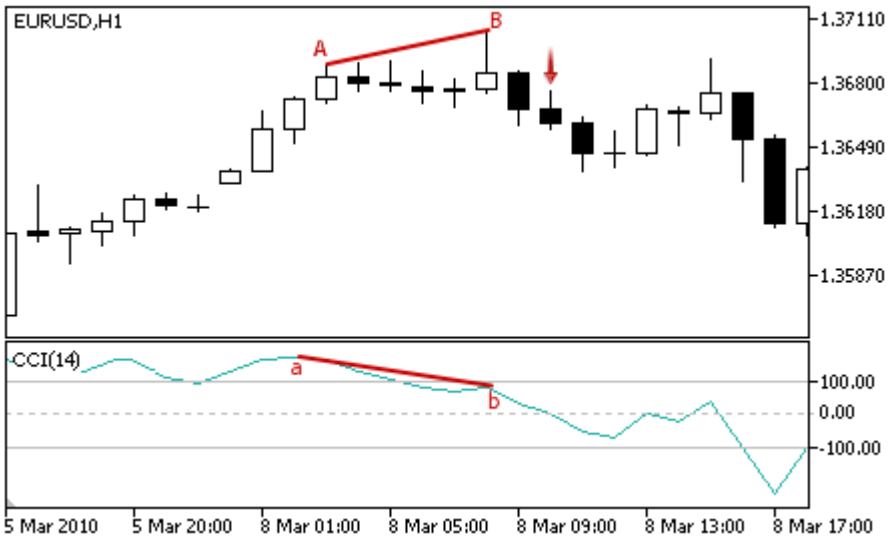
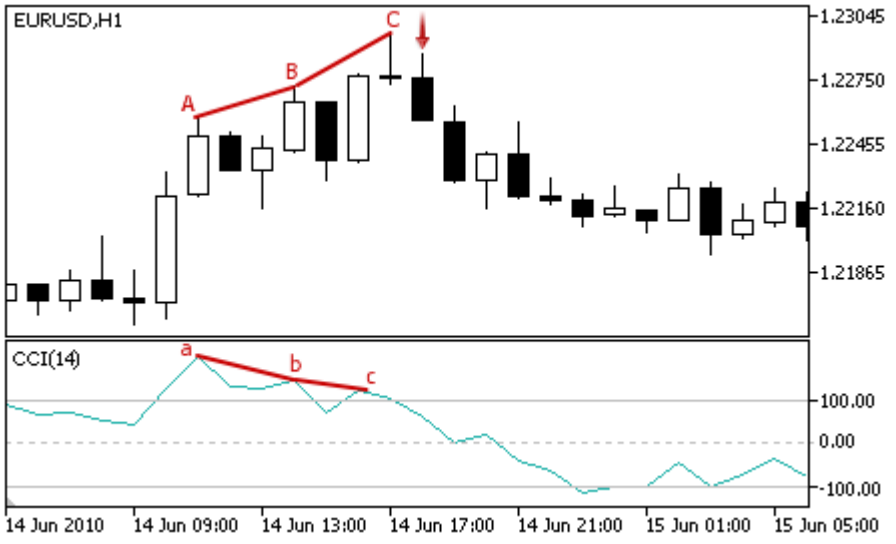
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **초과매도 수준 아래의 리버스** — 오실레이터가 위쪽으로 회전하고 분석된 막대의 값이 초과 매도 수준(기본값 -100)보다 낮습니다.  • **확산** — 첫 번째 분석된 오실레이터의 바닥은 이전 것보다 높고 해당 가격 바닥은 이전 것보다 낮습니다.  • **이중 확산** — 오실레이터는 세 개의 결과 바닥을 형성했고, 각각은 이전 것보다 높습니다; 그리고 가격은 세 개의 상응하는 바닥을 형성했고, 각각은 이전 것보다 낮습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• 초과매수 수준 아래의 리버스 — 오실레이터가 하향으로 전환되고 분석된 막대의 값이 초과매수 수준(기본값 100) 뒤에 있습니다.  • 확산 — 오실레이터의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 이중 확산 — 진동자는 세 개의 피크를 형성했고, 각각은 이전 피크보다 낮습니다. 그리고 가격은 세 개의 상응하는 피크를 형성했으며, 각각은 이전 피크보다 높습니다. 
매수하는 데에 이의 없음	오실레이터의 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	오실레이터 값은 분석된 막대에서 떨어집니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodCCI	오실레이터의 계산 주기.
Applied	오실레이터 계산에 사용되는 가격 시리즈입니다.

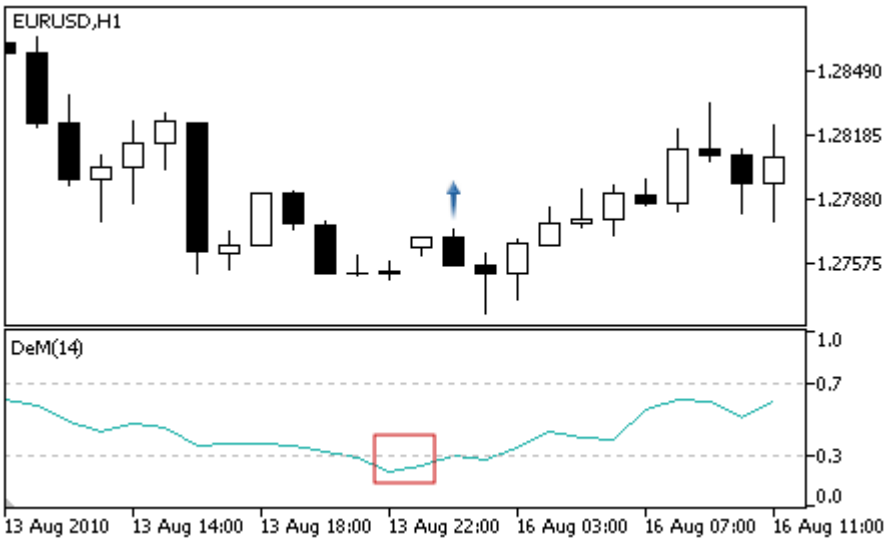
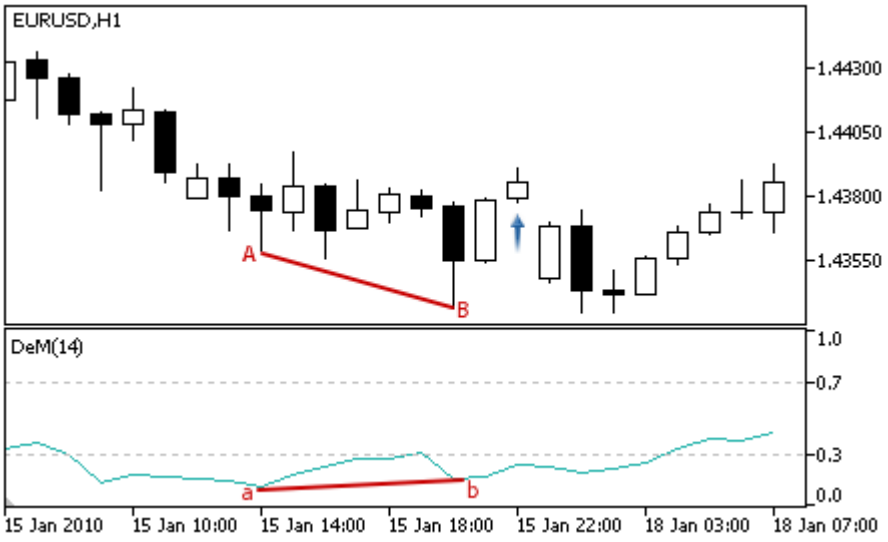
© 2000-2025, MetaQuotes Ltd.

오실레이터 DeMarker 시그널

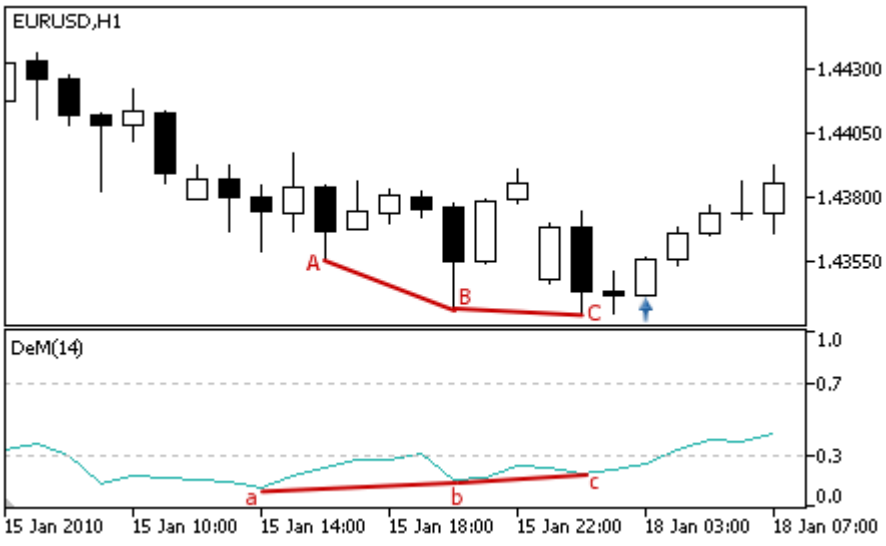
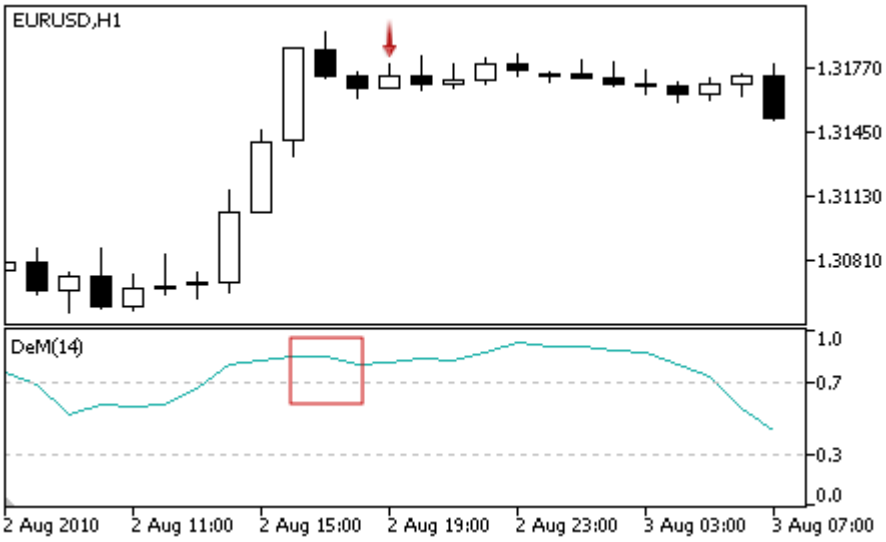
이 시그널 모듈은 오실레이터 DeMarker의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

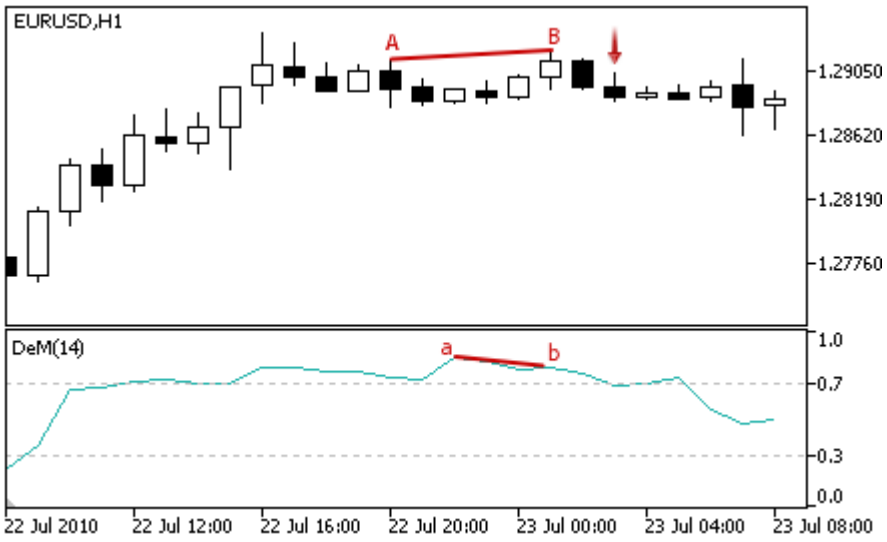
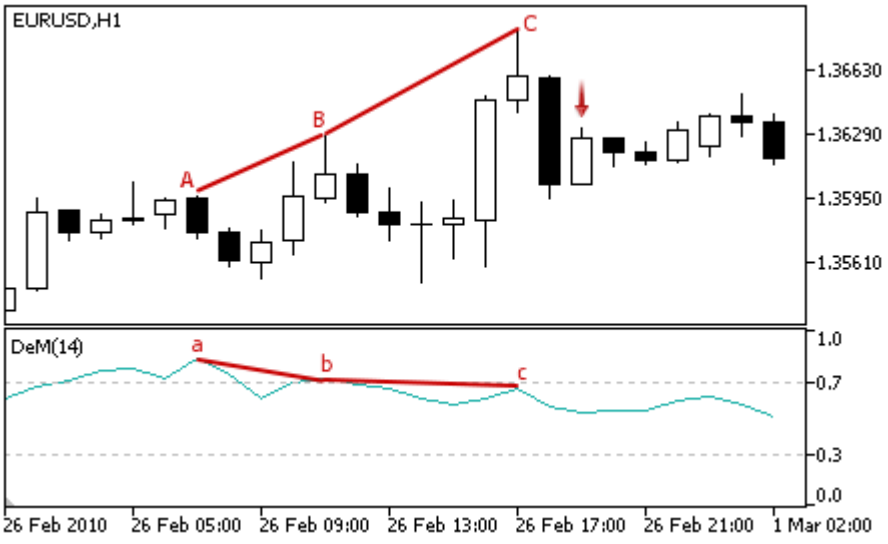
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **초과 매도 수준 아래의 리버스** —오실레이터가 위로 회전하고 분석 막대의 값이 초과 판매 수준 아래로 이동합니다(기본값 0.3).  • **확산** —첫 번째 분석된 오실레이터의 바닥은 이전 것보다 높고 해당 가격 바닥은 이전 것보다 낮습니다.  • **이중 확산** —오실레이터는 3개의 결과 바닥을 형성하고, 각각은 이전 바닥을 형성하고, 가격은 3개의 상응하는 바닥을 형성하고, 각각은 이전 것보다 낮습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• 초과 매수 수준 아래의 리버스 — 오실레이터가 하향으로 전환되고 분석된 막대의 값이 초과 매수 수준(기본값 0.7) 아래에 있습니다.  • 확산 — 오실레이터의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 이중 확산 —진동자는 세 개의 피크를 형성했고, 각각은 이전 피크보다 낮습니다. 그리고 가격은 세 개의 상응하는 피크를 형성했으며, 각각은 이전 피크보다 높습니다. 
매수하는 데에 이의 없음	오실레이터의 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	오실레이터 값은 분석된 막대에서 떨어집니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodDeM	오실레이터의 계산 주기.

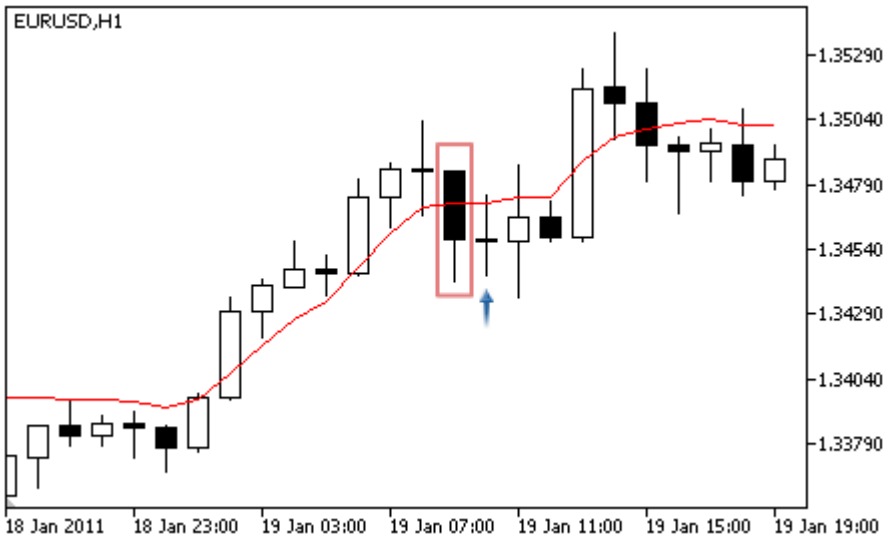
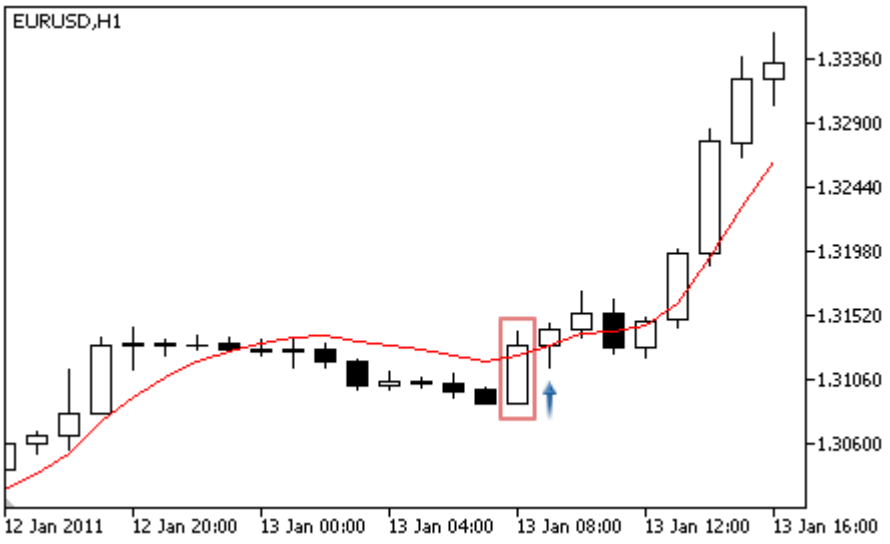
© 2000-2025, MetaQuotes Ltd.

이중 지수 이동 평균 지표 시그널

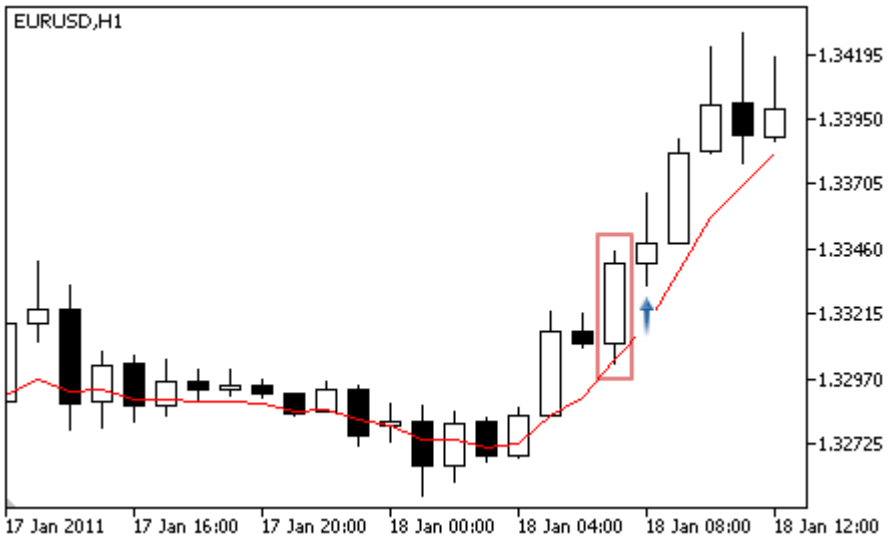
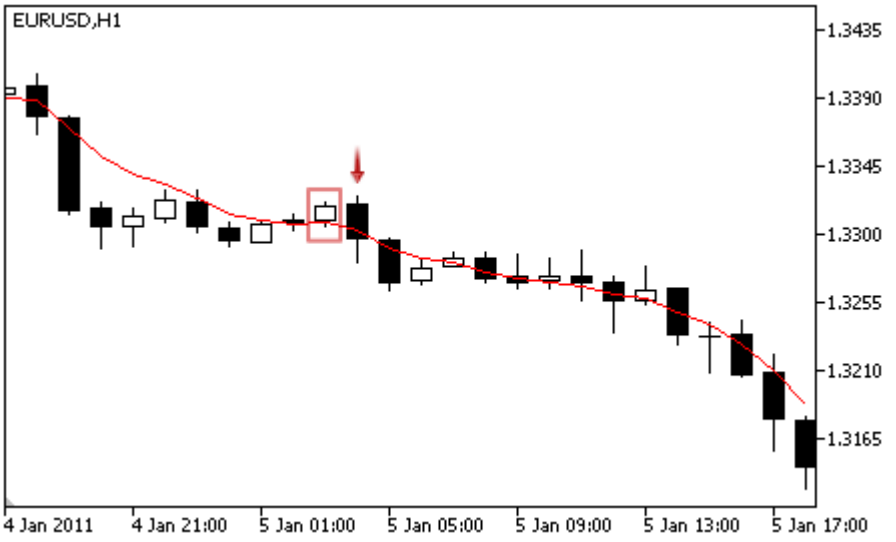
본 모듈은 이중 지수 이동 평균지표의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

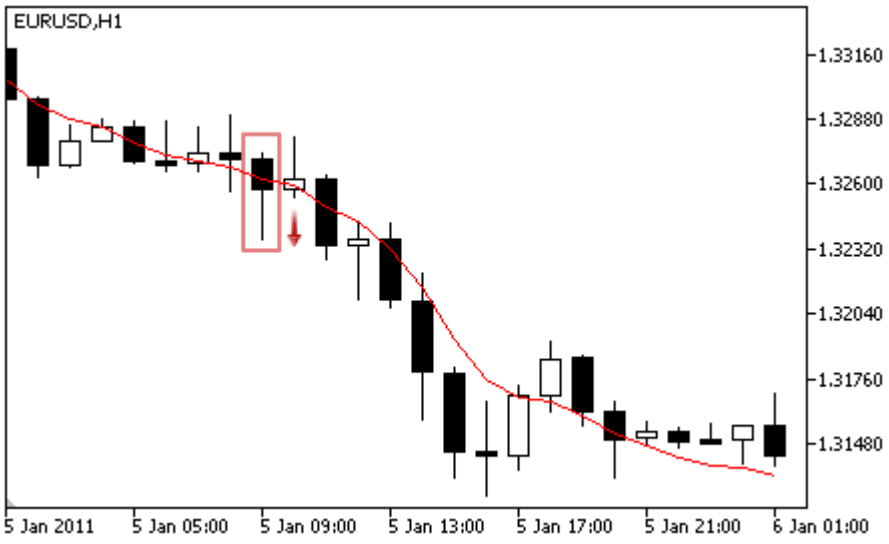
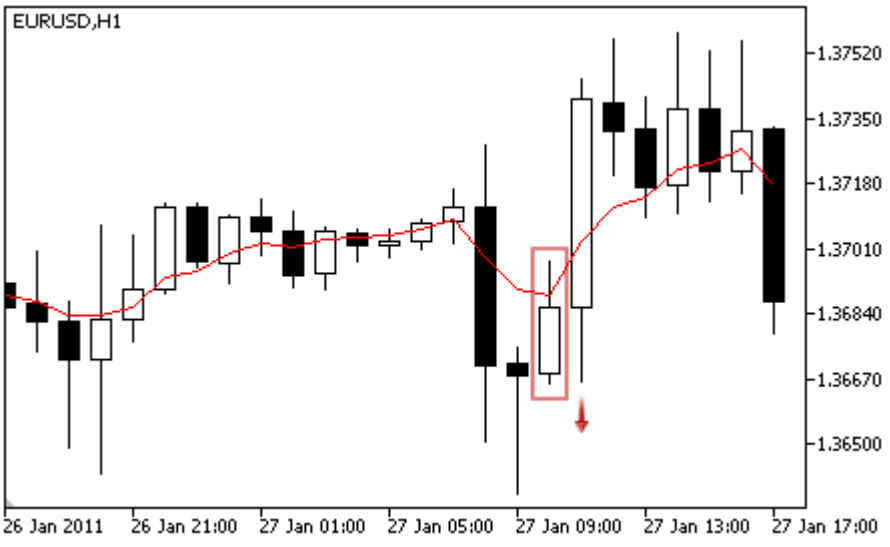
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **브레이크아웃 실패.** 가격이 지표 아래로 통과(분석된 막대의 시가는 지표 위에 있고 종가는 지표 아래에 있음)했으며 지표는 상승(지표 라인에서 롤백할 경우 약세 시그널)합니다.  • **이동 평균 교차.** 가격이 지표를 위로 통과했습니다(분석된 막대의 시가는 지표보다 낮고 종가는 지표보다 높습니다). 지표가 상승합니다(강세 시그널).  • **브레이크아웃 형성.** 막대 아래쪽 그림자가 지표(분석된 막대의 시가 및 종가가 지표 위에 있고, 저가는 지표 아래 있음)와 지표(지표 라인에서 롤백 시그널)를 교차했습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• **브레이크아웃 실패**. 가격이 지표를 위쪽으로 통과했습니다(분석된 막대의 시가는 지표 아래에 있고 종가는 지표 위에 있음). 지표는 하락합니다(지표 라인에서 롤백할 경우 약세 시그널).  • **이동 평균 교차**. 가격이 지표를 하향(분석된 막대의 시가는 지표보다 높고 종가는 지표보다 낮음)하고 지표는 하락(강세 시그널)합니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 브레이크아웃 형성. 바의 위쪽 그림자가 지표(분석된 바의 시가 및 종가 지표 아래에 있고, 고가가 지표 위에 있음)을 통과하고 지표는 하락합니다(지표 라인에서 롤백할 경우 약세 시그널). 
매수하는 데에 이의 없음	가격이 지표보다 위에 있습니다.
매도하는 데에 이의 없음	가격이 지표보다 아래에 있습니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodMA	지표의 평균화 기간.
Shift	시간 축을 따라 지표를 이동합니다(막대).
Method	평균화 메서드.
Applied	지표 계산에 사용되는 가격 시리즈.

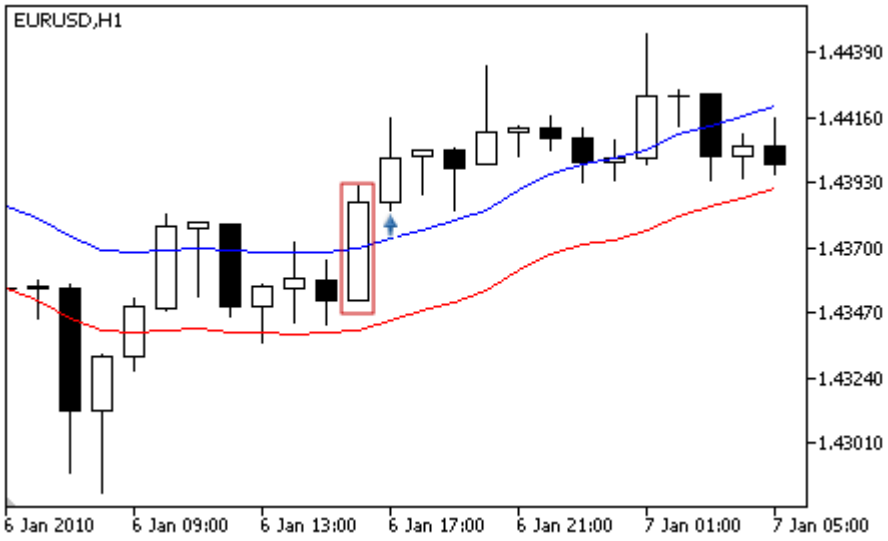
© 2000-2025, MetaQuotes Ltd.

Envelopes 지표 시그널

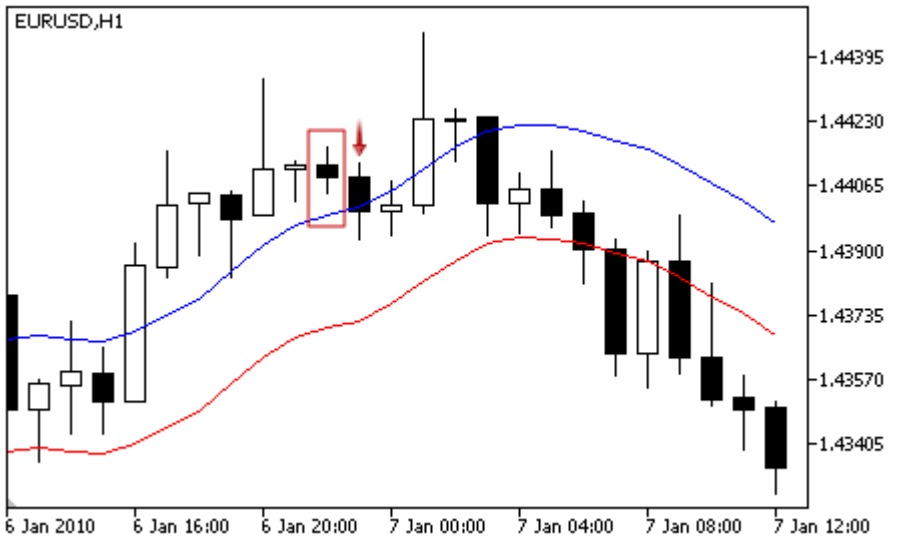
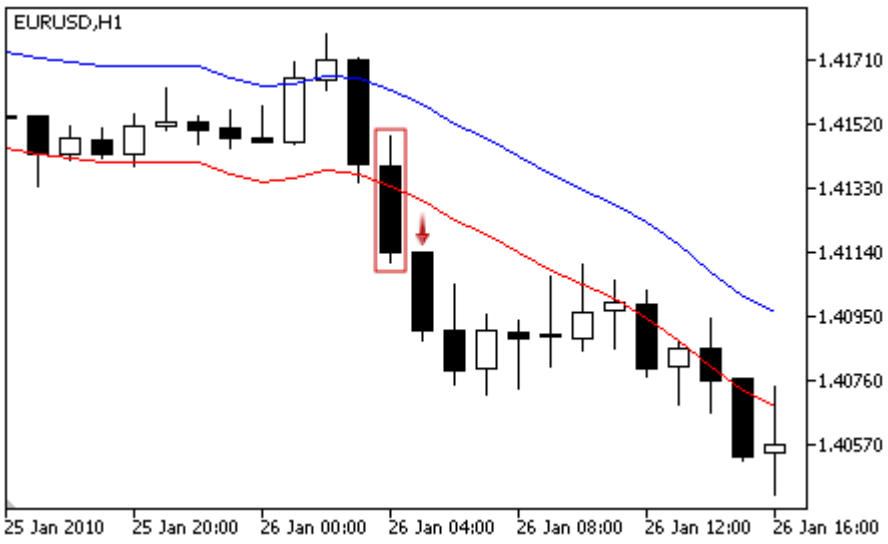
이 시그널 모듈은 지표 Envelopes의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• 가격은 분석된 막대의 지표 하단선 근처입니다. • 가격은 분석된 막대의 지표 상단을 넘어섰습니다. 
매도용	• 가격은 분석된 막대에서 지표의 윗선에 가깝습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 가격은 분석된 막대의 지표 하한선을 넘었습니다. 
매수하는 데에 이의 없음	시그널 없음.
매도하는 데에 이의 없음	시그널 없음.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodMA	지표의 계산 기간.
Shift	시간 축을 따라 지표를 이동합니다(막대).
Method	평균화 메서드.
Applied	지표 계산에 사용되는 가격 시리즈.
Deviation	중심선(MA)으로부터의 envelope 경계 편차(백분율).

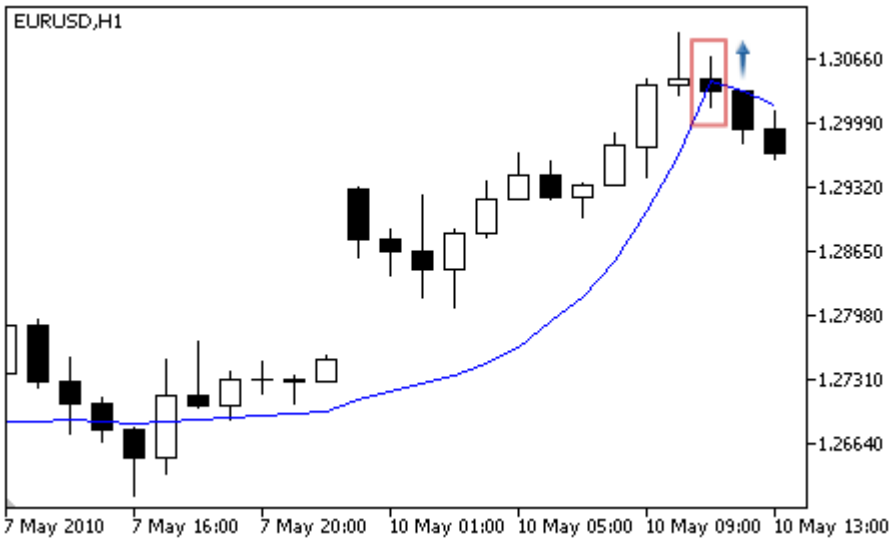
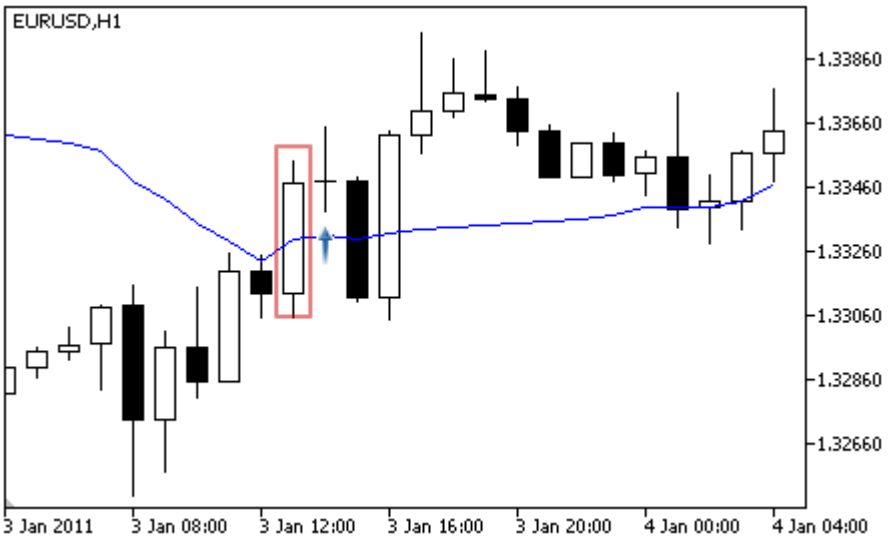
© 2000-2025, MetaQuotes Ltd.

프랙탈 적응형 이동평균 지표 시그널

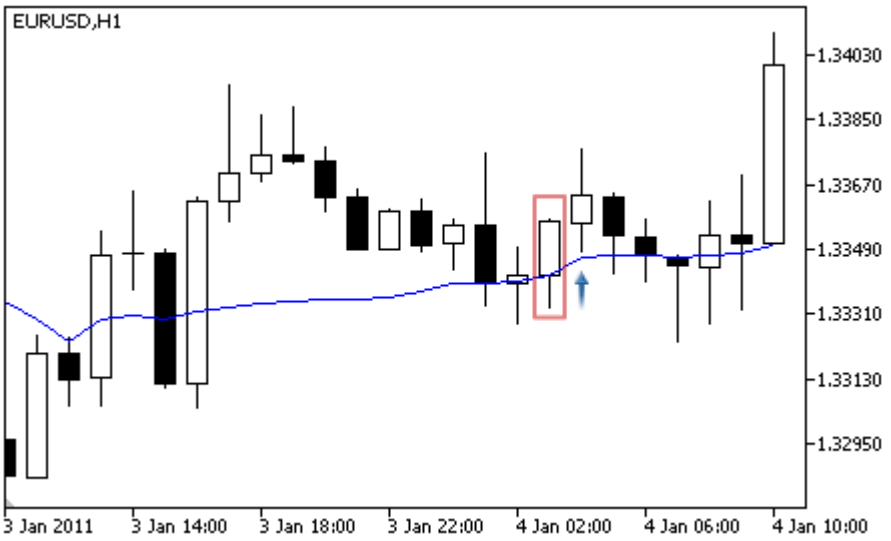
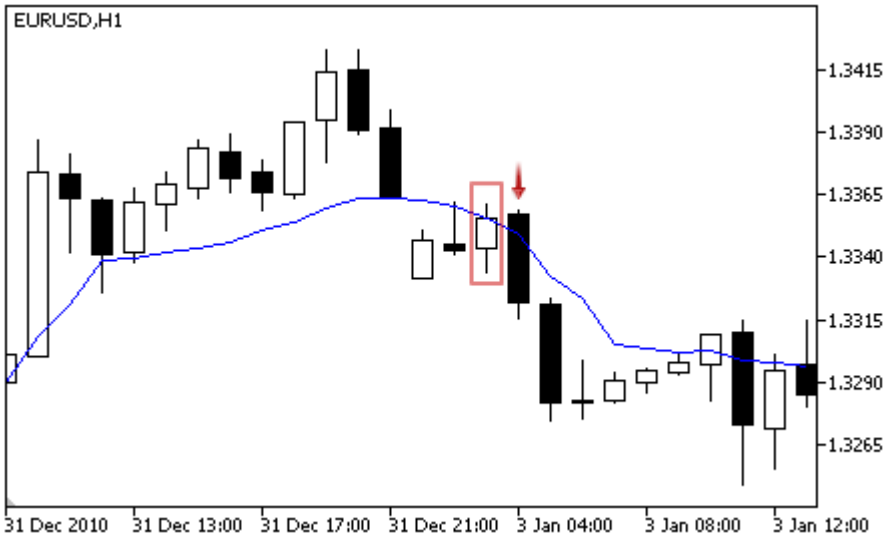
이 시그널 모듈은 프랙탈 적응 이동 평균 지표의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

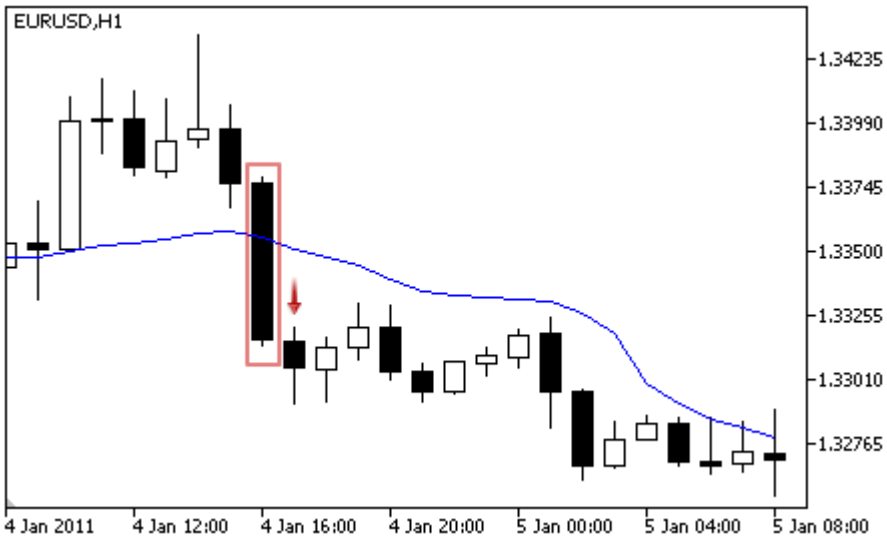
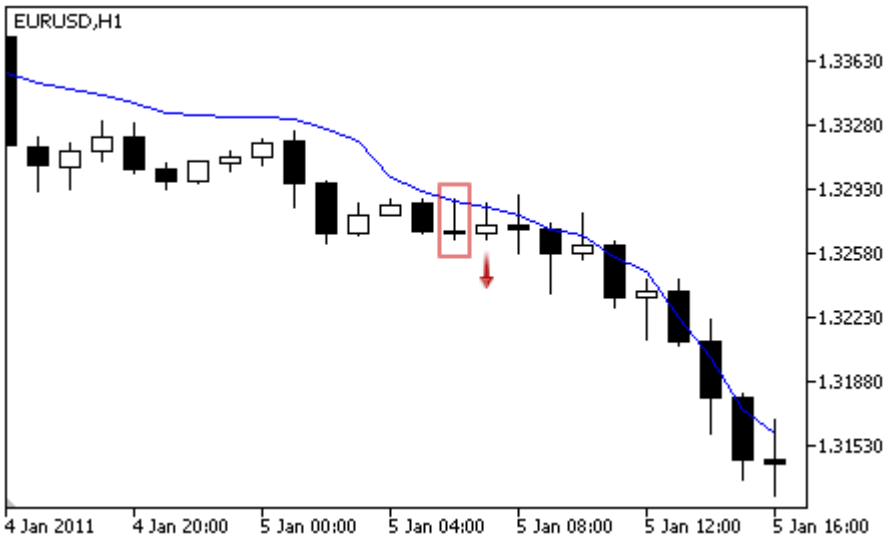
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **브레이크아웃 실패.** 가격이 지표 아래로 통과(분석된 막대의 시가는 지표 위에 있고 종가는 지표 아래에 있음)했으며 지표는 상승(지표 라인에서 롤백할 경우 약세 시그널)합니다.  • **이동 평균 교차.** 가격이 지표를 위로 통과했습니다(분석된 막대의 시가는 지표보다 낮고 종가는 지표보다 높습니다). 지표가 상승합니다(강세 시그널). 

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	• **브레이크아웃 형성.** 막대의 아래쪽 그림자가 지표(분석된 막대의 시가 및 종가가 지표 위에 있고 저가는 지표 아래에 있음)를 통과하고 지표가 상승합니다(지표 라인에서 롤백 신호). 
매도용	• **브레이크아웃 실패.** 가격이 지표를 위쪽으로 통과했습니다(분석된 막대의 시가는 지표 아래에 있고 종가는 지표 위에 있음). 지표는 하락합니다(지표 라인에서 롤백할 경우 약세 시그널).  • **이동 평균 교차.** 가격이 지표를 하향(분석된 막대의 시가는 지표보다 높고 종가는 지표보다 낮음)하고 지표는 하락(강세 시그널)합니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 브레이크아웃 형성. 바의 위쪽 그림자가 지표(분석된 바의 시가 및 종가 지표 아래에 있고, 고가가 지표 위에 있음)을 통과하고 지표는 하락합니다(지표 라인에서 롤백할 경우 약세 시그널). 
매수하는 데에 이의 없음	가격이 지표보다 위에 있습니다.
매도하는 데에 이의 없음	가격이 지표보다 아래에 있습니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodMA	지표의 평균화 기간.
Shift	시간 축을 따라 지표를 이동합니다(막대).
Method	평균화 메서드.
Applied	지표 계산에 사용되는 가격 시리즈.

© 2000-2025, MetaQuotes Ltd.

하루 중 시간 필터 시그널

본 모듈은 시장 모델의 효율성이 시간에 따라 변화한다는 가정에 기초합니다. 이 모듈을 사용하여 다른 모듈에서 수신한 신호를 시간 및 요일별로 필터링할 수 있습니다. 이로 인해 불리한 시간을 차단하여 생성된 시그널의 품질을 높일 수 있습니다. 모듈의 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	시그널 없음.
매도용	시그널 없음.
매수하는 데에 이의 없음	현재 날짜 및 시간이 지정된 매개 변수를 충족합니다.
매도하는 데에 이의 없음	현재 날짜 및 시간이 지정된 매개 변수를 충족합니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
GoodHourOfDay	거래 시그널이 활성화될 하루 중 유일한 시간(0-23)입니다. 값이 -1이면 시그널이 하루 종일 활성화됩니다.
BadHoursOfDay	비트 필드. 이 필드의 각 비트는 하루 중 한 시간(0비트 - 0시간, ..., 23비트 - 23번째 시간)에 해당합니다. 비트 값이 0이면 해당 시간 동안 트레이드 시그널이 활성화됩니다. 비트 값이 1이면 해당 시간 동안 트레이드 시그널이 비활성화됩니다. 지정된 숫자는 이진수로 표시되며 비트 마스크로 사용됩니다. 비활성화된 시간은 활성화된 시간보다 우선 순위가 높습니다.
GoodDayOfWeek	거래 시그널이 활성화될 유일한 요일(0～6일, 여기서 0은 일요일)의 수입니다. 값이 -1이면 신호는 언제든지 활성화됩니다.
BadDaysOfWeek	비트 필드. 이 필드의 각 비트는 요일(0비트 - 일요일, ..., 6비트 - 토요일)에 해당합니다. 비트 값이 0이면 해당 일 동안 트레이드 시그널이 활성화됩니다. 비트 값이 1이면 해당 일 동안 트레이드 시그널이 비활성화됩니다. 지정된 숫자는 이진수로 표시되며 비트 마스크로 사용됩니다. 비활성화된 요일은 활성화된 요일보다 우선 순위가 높습니다.

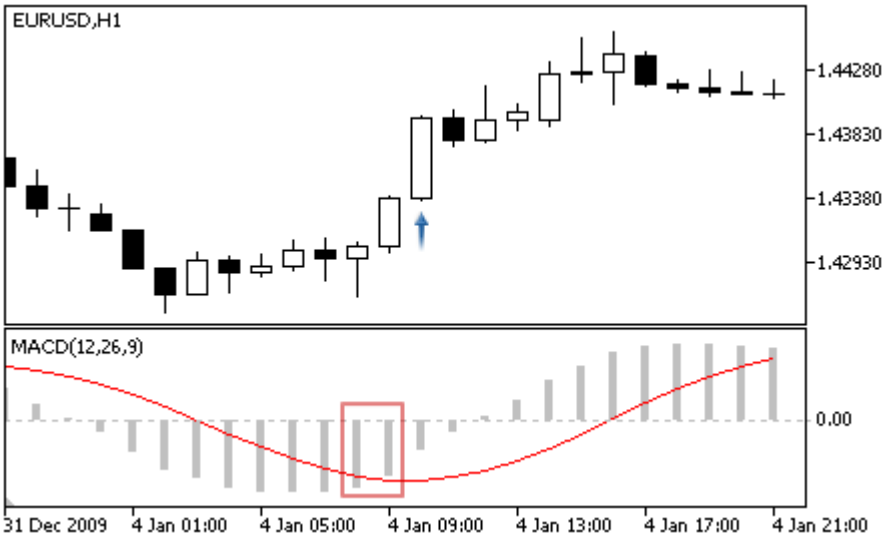
© 2000-2025, MetaQuotes Ltd.

Oscillator MACD 시그널

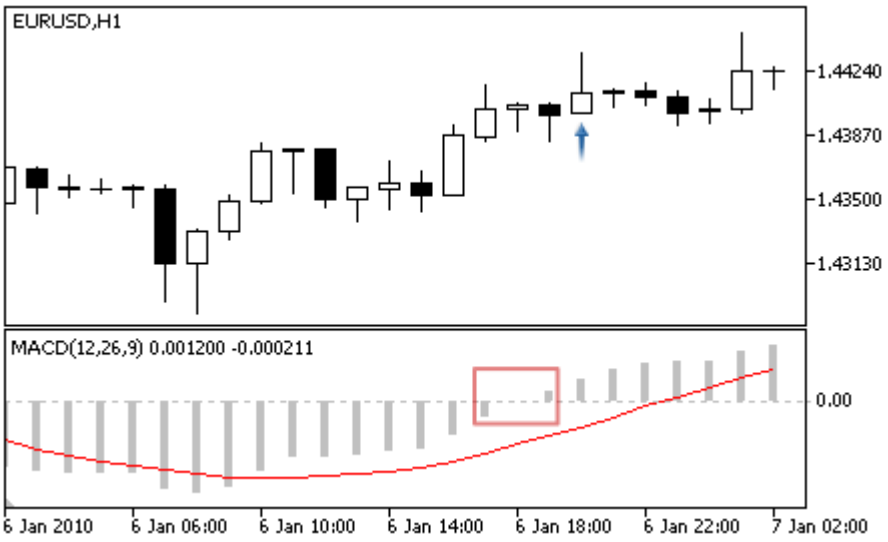
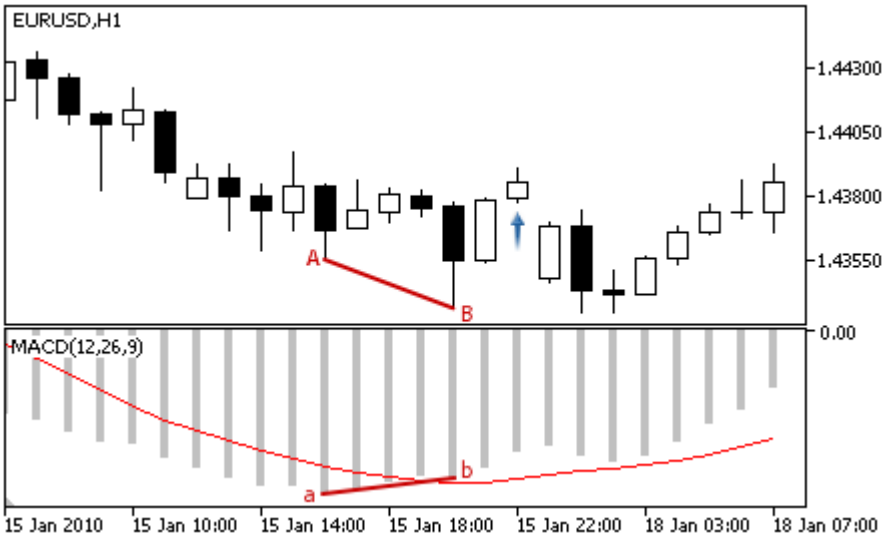
이 시그널 모듈은 오실레이터 MACD의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

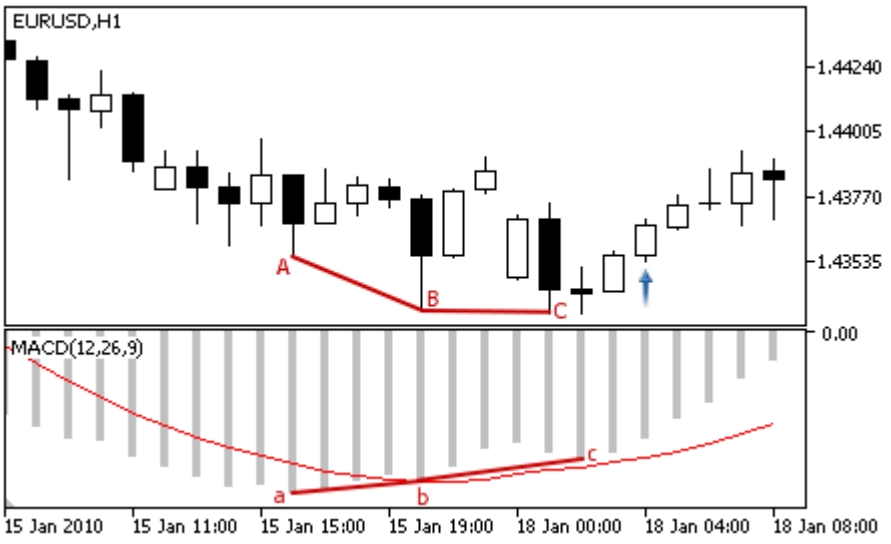
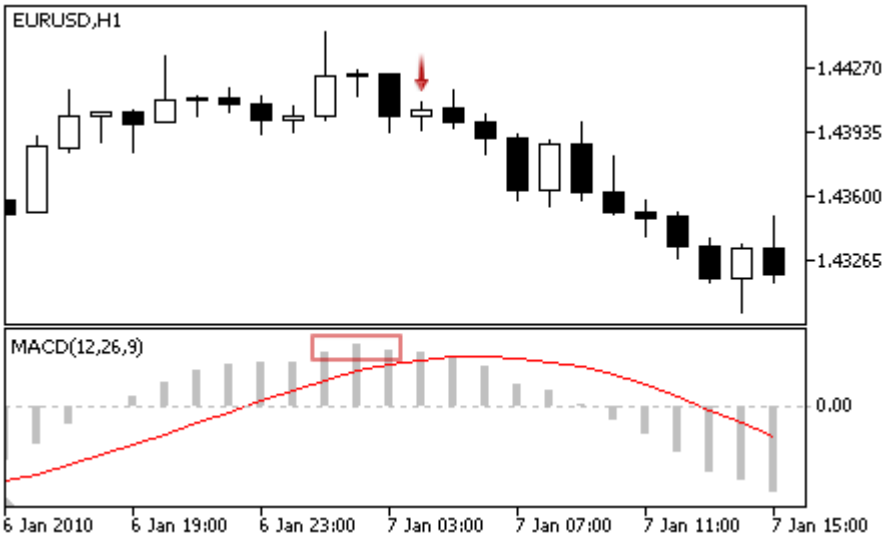
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **리버스** —오실레이터가 위쪽으로 회전했음(오실레이터는 분석된 막대에서 상승하고 이전 막대에서 하락함). • **메인 선과 시그널 선의 교차** —메인 라인이 분석된 바의 신호 라인 위 및 이전 신호 라인 아래에 있습니다.  • **영점 교차** —메인 라인은 분석된 바에서는 0 레벨 위에 있으며 이전 바에서는 0 레벨 아래에 있습니다.

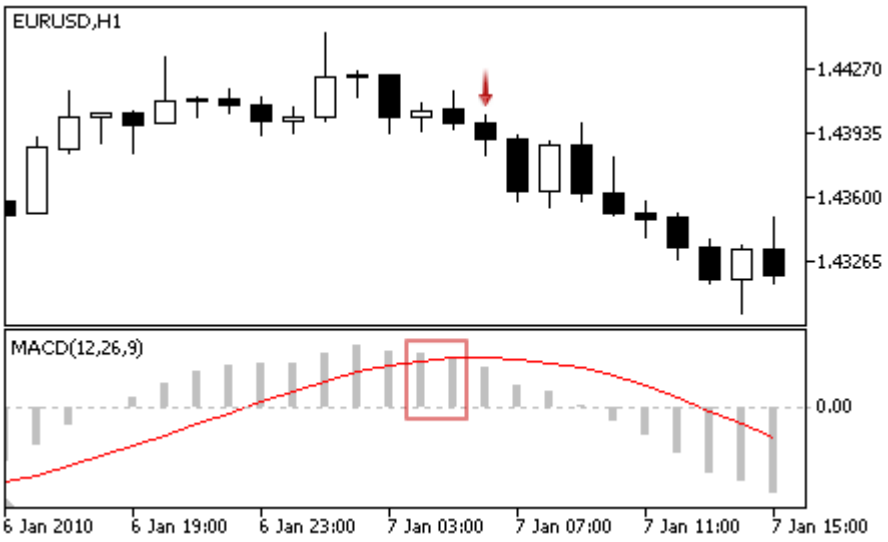
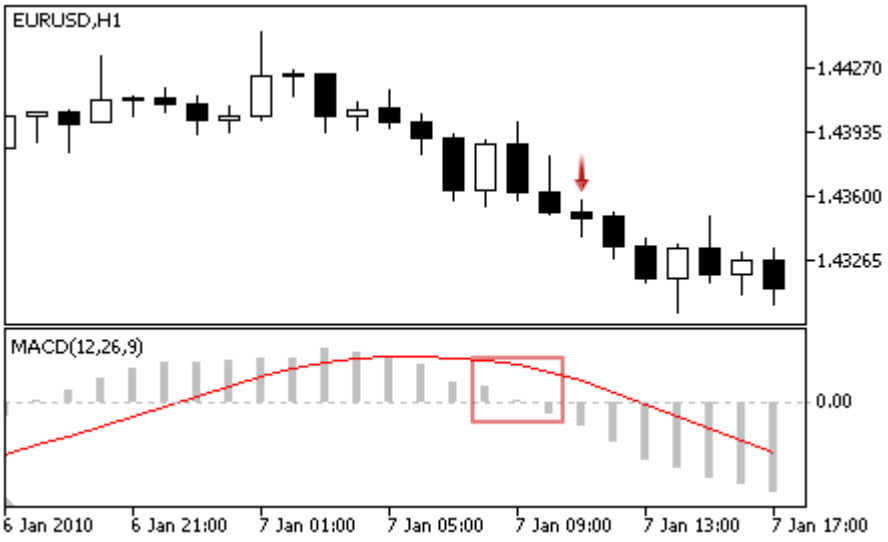
© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 확산 — 첫 번째 분석된 오실레이터의 바닥은 이전 것보다 높고 해당 가격 바닥은 이전 것보다 낮습니다.  • 이중 확산 — 오실레이터는 3개의 결과 바닥을 형성하고, 각각은 이전 것보다 높으며, 가격은 3개의 상응하는 바닥을 형성하고, 각각은 이전 것보다 낮습니다.

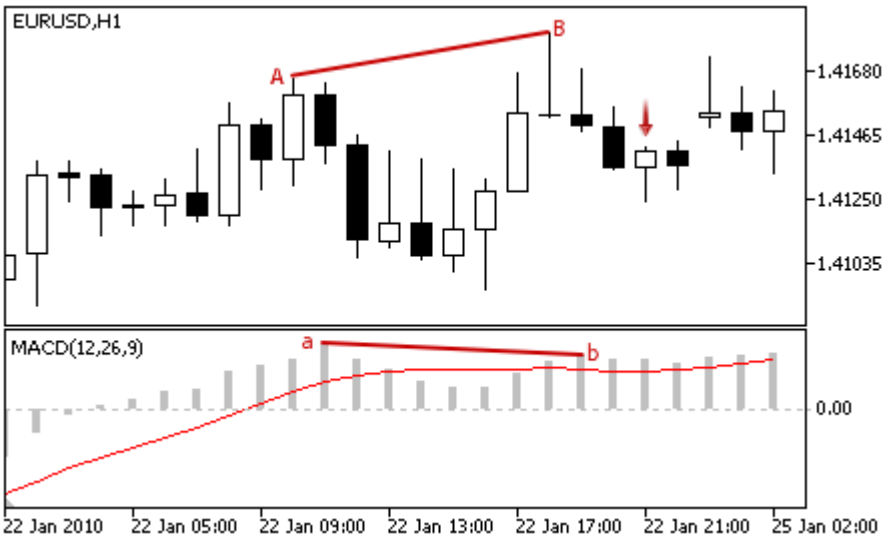
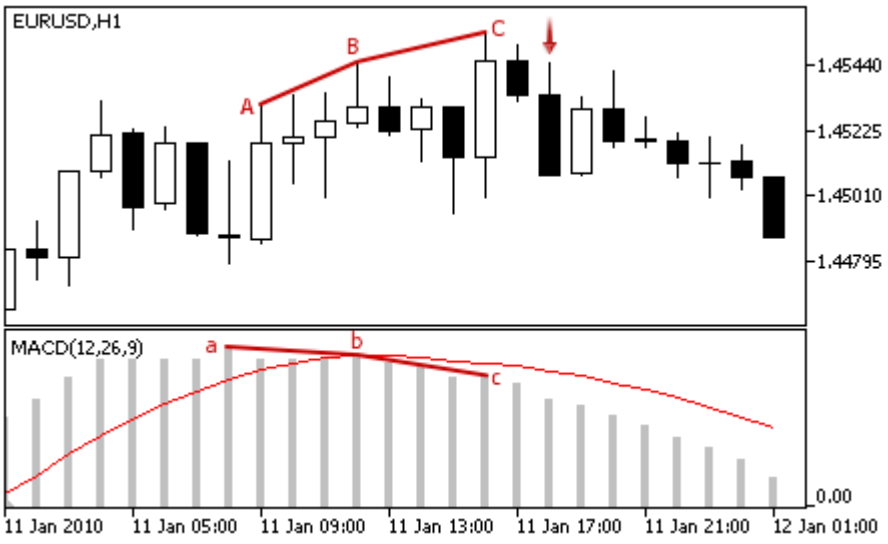
© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• 리버스 — 오실레이터가 아래쪽으로 향했습니다(오실레이터는 분석된 막대에 떨어지며 이전 막대에 상승합니다.  • 메인선과 시그널 선의 교차 — 메인 라인이 분석된 막대의 시그널 라인 아래 및 이전 막대의 시그널 라인 위에 있습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 영점 교차 — 메인 라인은 분석된 막대에서는 영점 이하이고 이전 바에서는 영점 이상입니다.  • 확산 — 오실레이터의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 이중 확산 —진동자는 세 개의 피크를 형성했고, 각각은 이전 피크보다 낮습니다. 그리고 가격은 세 개의 상응하는 피크를 형성했으며, 각각은 이전 피크보다 높습니다. 
매수하는 데에 이의 없음	오실레이터의 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	오실레이터 값은 분석된 막대에서 떨어집니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodFast	빠른 EMA의 계산 기간.
PeriodSlow	느린 EMA의 계산 기간.
PeriodSignal	평활 기간.
Applied	오실레이터 계산에 사용되는 가격 시리즈입니다.

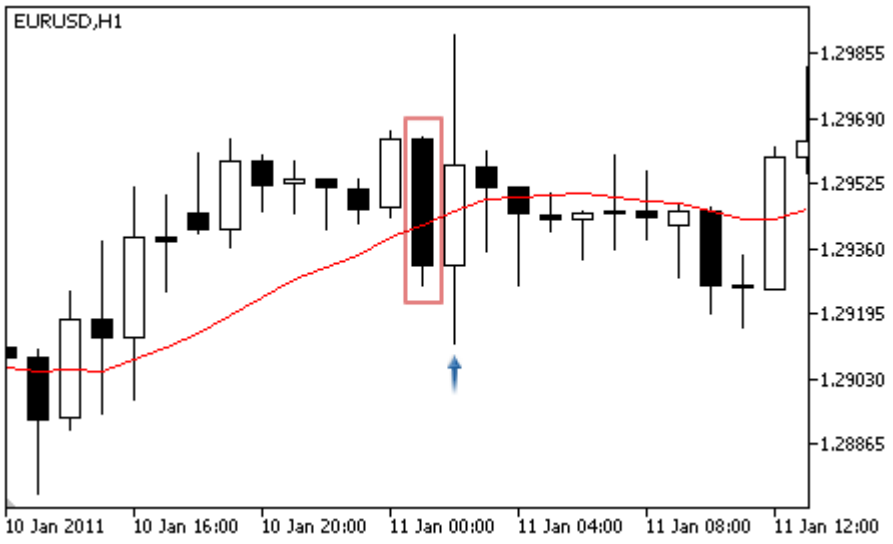
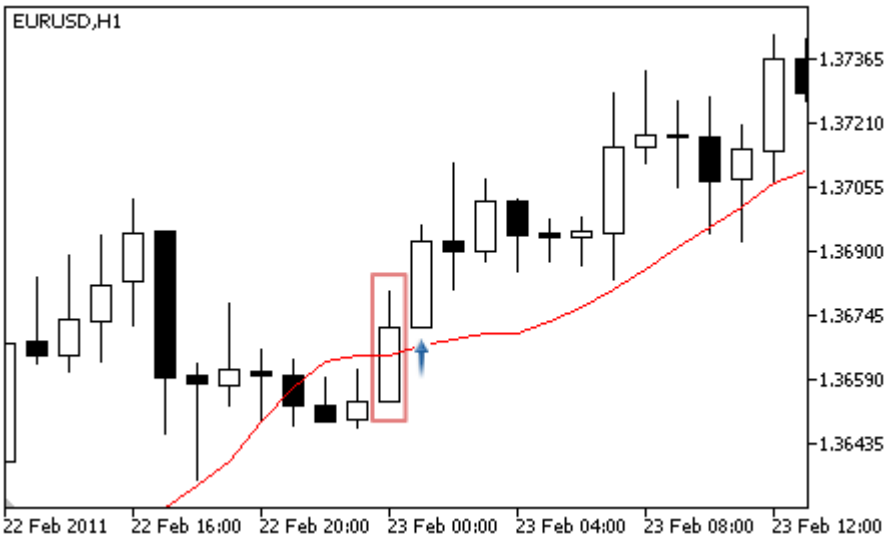
© 2000-2025, MetaQuotes Ltd.

이동 평균 지표 시그널

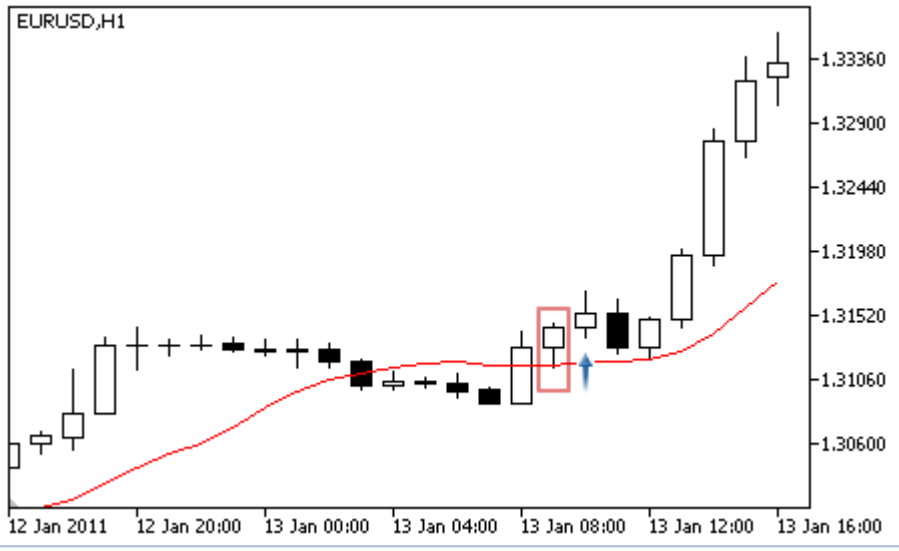
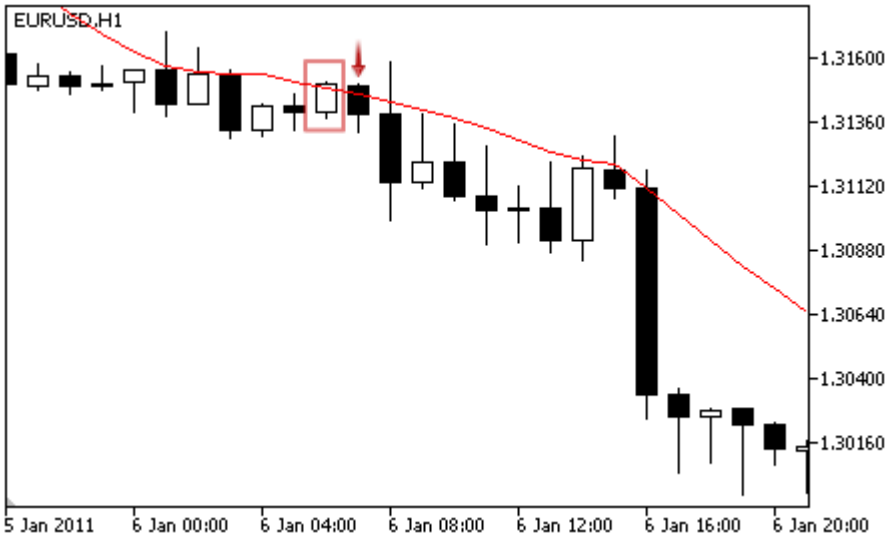
이 시그널 모듈은 이동 평균 지표의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

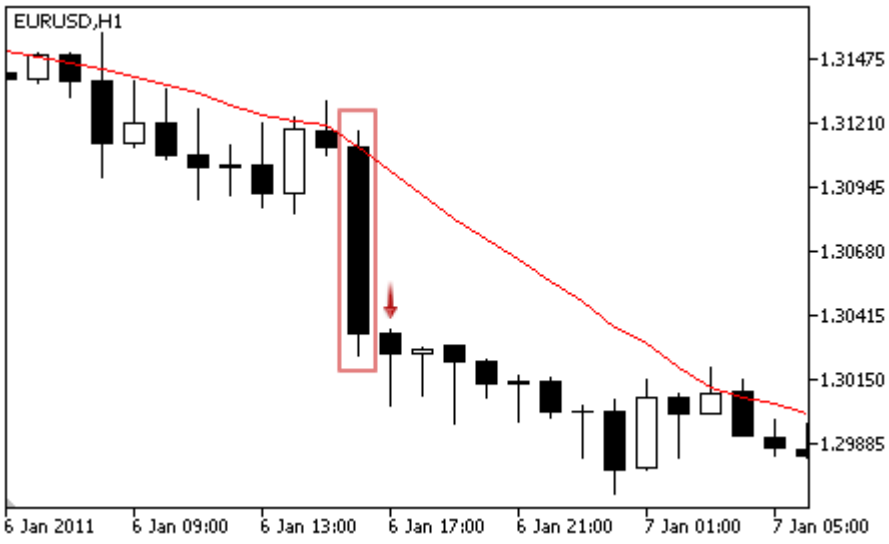
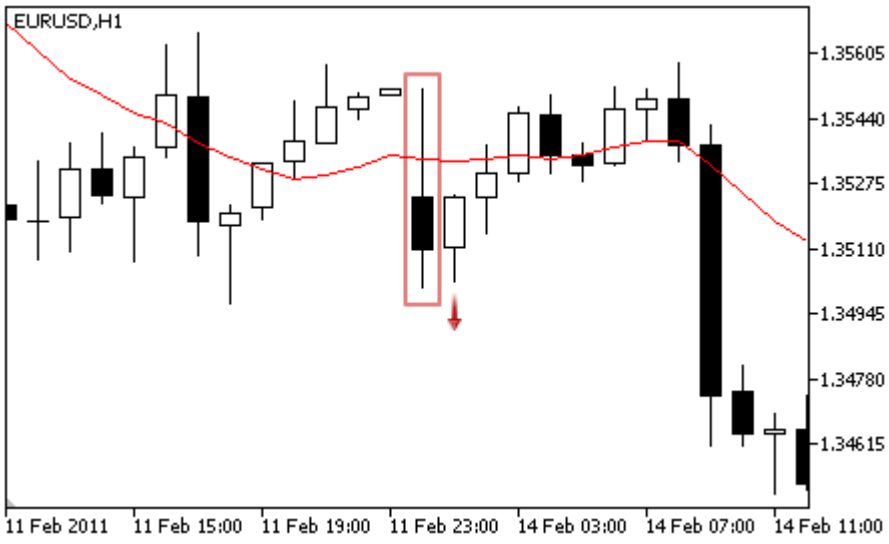
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **브레이크아웃 실패**. 가격이 지표 아래로 통과(분석된 막대의 시가는 지표 위에 있고 종가는 지표 아래에 있음)했으며 지표는 상승(지표 라인에서 롤백할 경우 약세 시그널)합니다.  • **이동 평균 교차**. 가격이 지표를 위로 통과했습니다(분석된 막대의 시가는 지표보다 낮고 종가는 지표보다 높습니다). 지표가 상승합니다(강세 시그널).  • **브레이크아웃 형성**. 막대의 아래쪽 그림자가 지표(분석된 막대의 시가 및 종가는 지표 위에 있고 저가는 지표 아래에 있음)을 통과하고 지표가 상승합니다(지표 라인에서 롤백을 위한 약세 시그널).

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• 브레이크아웃 실패. 가격이 지표를 위로 통과했으며(분석된 막대의 시가가 지표보다 낮고 종가가 지표보다 높음) 지표는 하락합니다(약세 신호).  • 이동 평균 교차. 가격이 지표를 하향(분석된 막대의 시가는 지표보다 높고 종가는 지표보다 낮음)하고 지표는 하락(강세 시그널)합니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 브레이크아웃 형성. 바의 위쪽 그림자가 지표(분석된 바의 시가 및 종가 지표 아래에 있고, 고가가 지표 위에 있음)을 통과하고 지표는 하락합니다(지표 라인에서 롤백할 경우 약세 시그널). 
매수하는 데에 이의 없음	가격이 지표보다 위에 있습니다.
매도하는 데에 이의 없음	가격이 지표보다 아래에 있습니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodMA	지표의 평균화 기간.
Shift	시간 축을 따라 지표를 이동합니다(막대).
Method	평균화 메서드.
Applied	지표 계산에 사용되는 가격 시리즈.

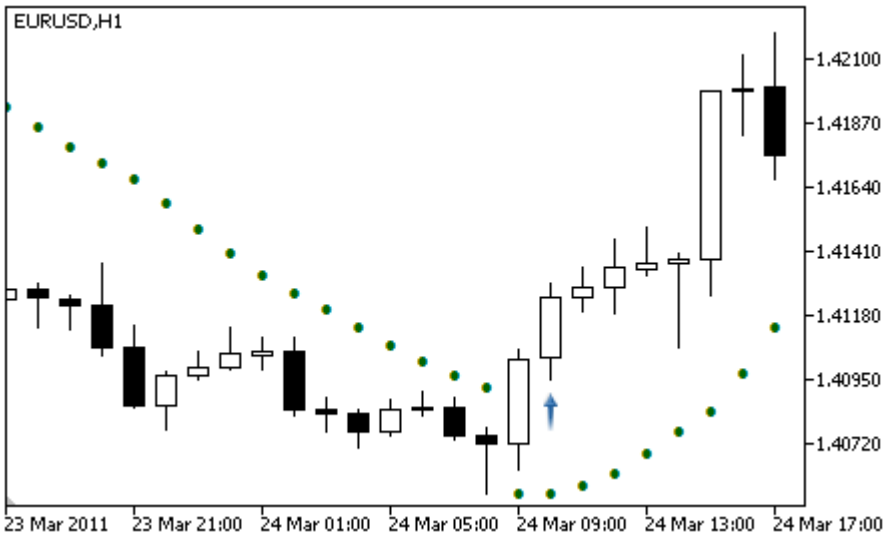
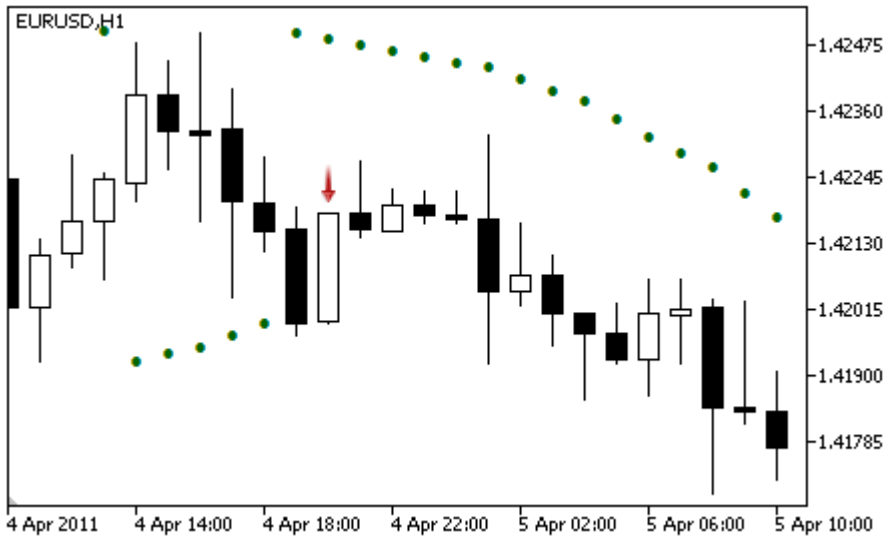
© 2000-2025, MetaQuotes Ltd.

파라볼릭 SAR 지표 시그널

이 시그널 모듈은 지표 파라볼릭 SAR의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	리버스 — 지표는 분석된 막대의 가격보다 낮고 이전 막대의 가격보다 높습니다. 
매도용	리버스 — 지표는 분석된 막대의 가격보다 높고 이전 막대의 가격보다 낮습니다. 
매수하는 데에 이의 없음	가격이 지표보다 위에 있습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
매도하는 데에 이의 없음	가격이 지표보다 아래에 있습니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
Step	가격 인상 단계.
Maximum	지표와 가격의 수렴 속도 최대 속도.

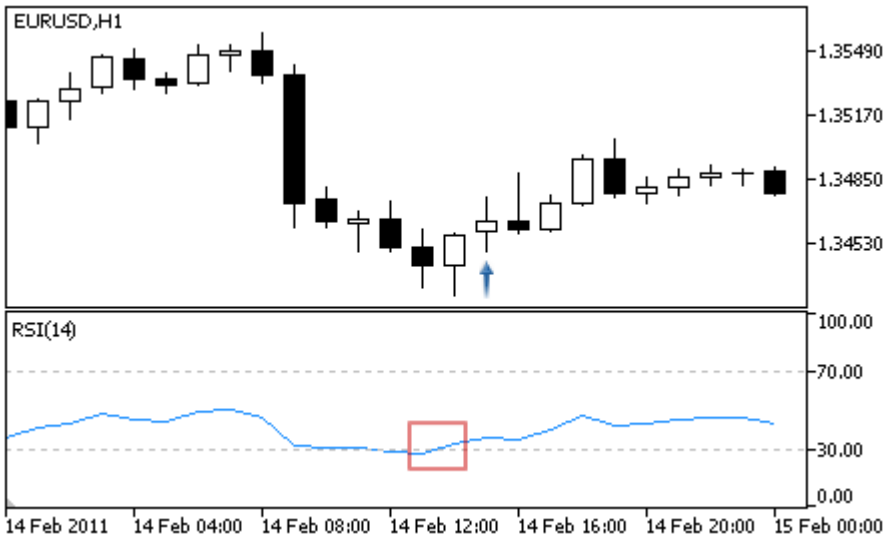
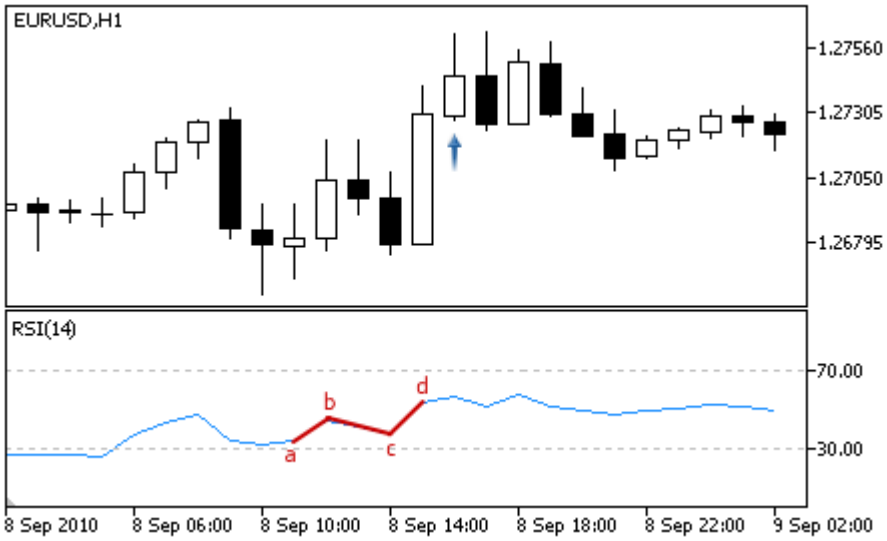
© 2000-2025, MetaQuotes Ltd.

오실레이터 상대적 강도 지수 시그널

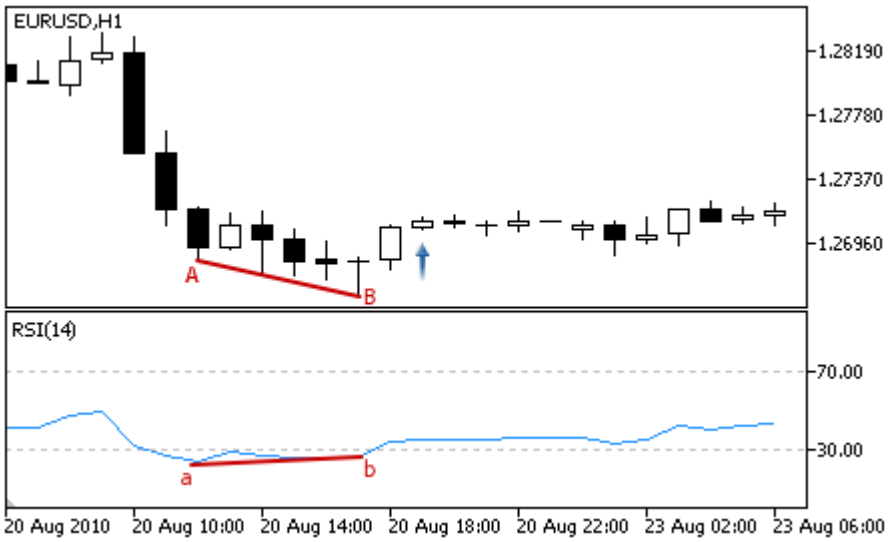
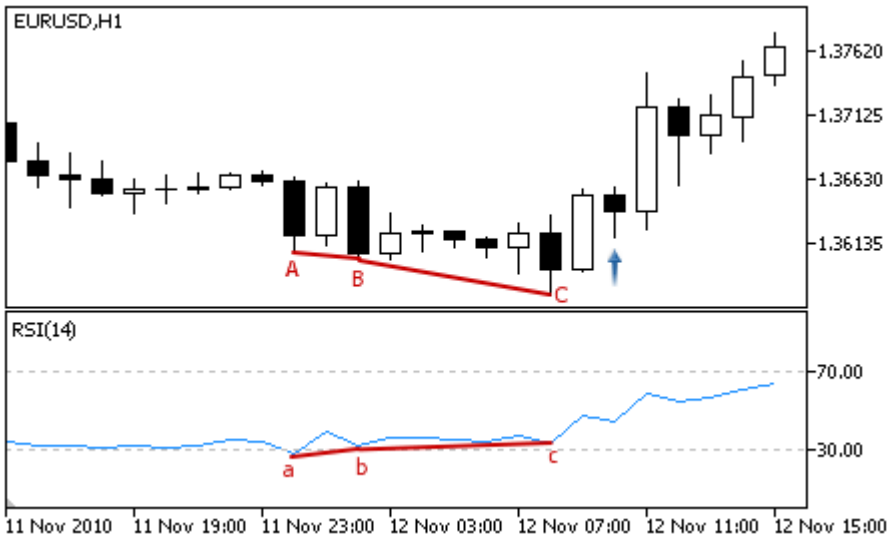
이 시그널 모듈은 오실레이터 상대 강도 지수의 시장 모델에 기초합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

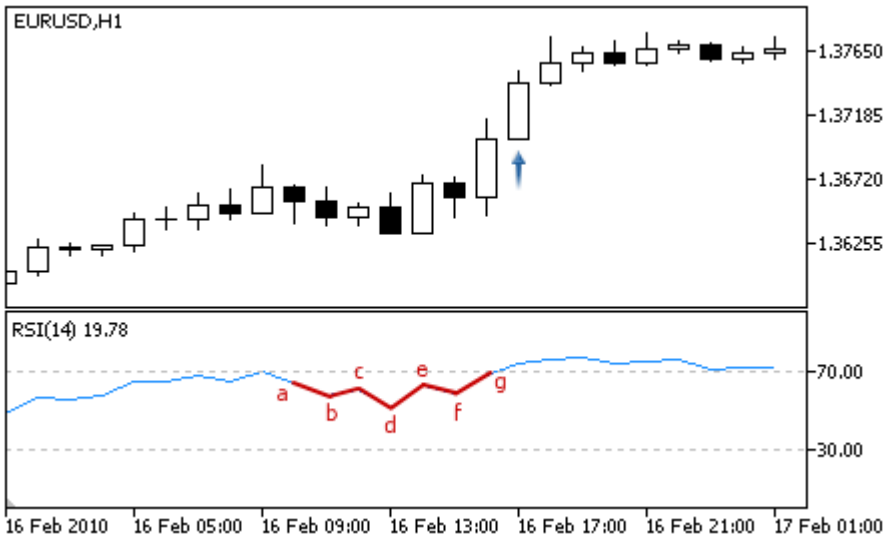
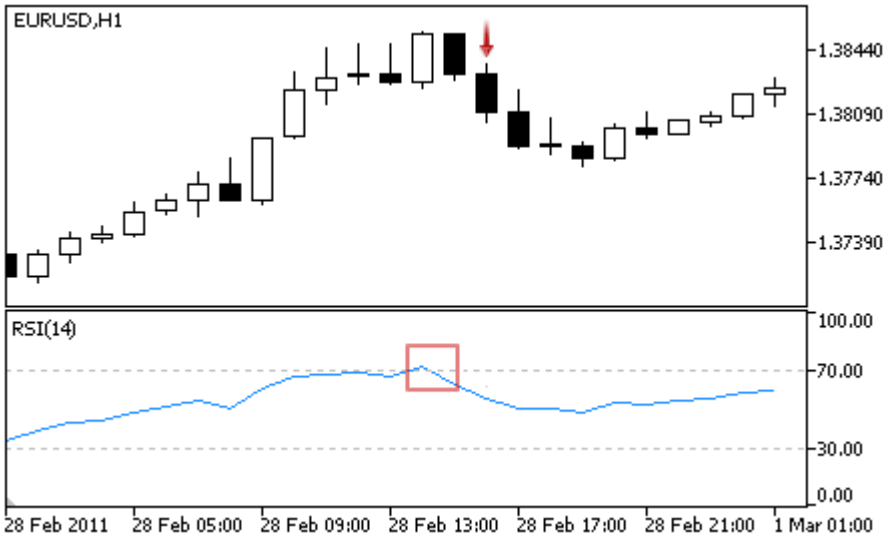
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• 초과매도 수준 아래의 리버스 — 오실레이터가 위쪽으로 회전하고 분석된 바의 값이 초과 매도 수준(기본값 30)에 미치지 않습니다.  • 스윙 실패 — 오실레이터가 분석 막대의 이전 피크보다 높게 상승합니다.  • 확산 — 첫 번째 분석된 오실레이터의 바닥은 이전 것보다 낮고 해당 가격 바닥은 이전 것보다 낮습니다.

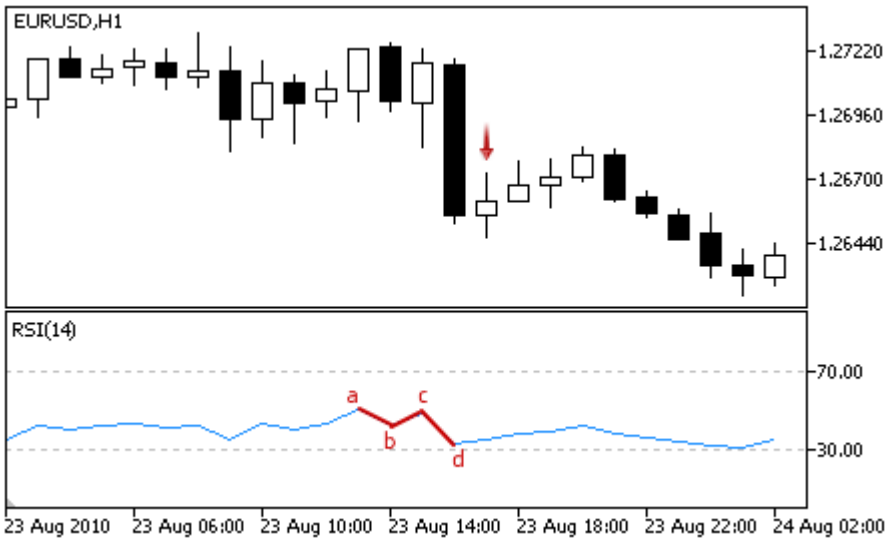
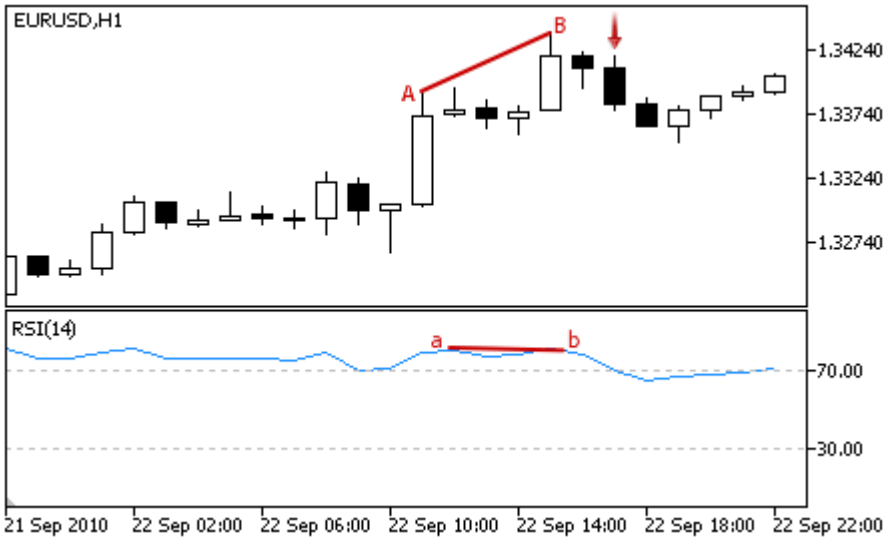
© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • **이중 확산** —오실레이터는 3개의 결과 바닥을 형성하고, 각각은 이전 바닥을 형성하고, 가격은 3개의 상응하는 바닥을 형성하고, 각각은 이전 것보다 낮습니다.  • **Head/Shoulders** —오실레이터는 세 개의 결과 바닥을 형성하고, 중간 하나는 다른 것들보다 낮습니다.

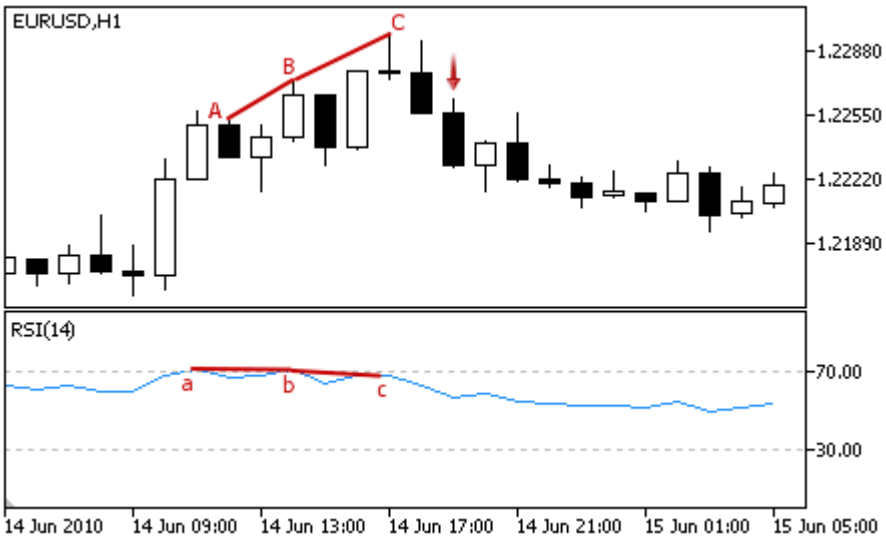
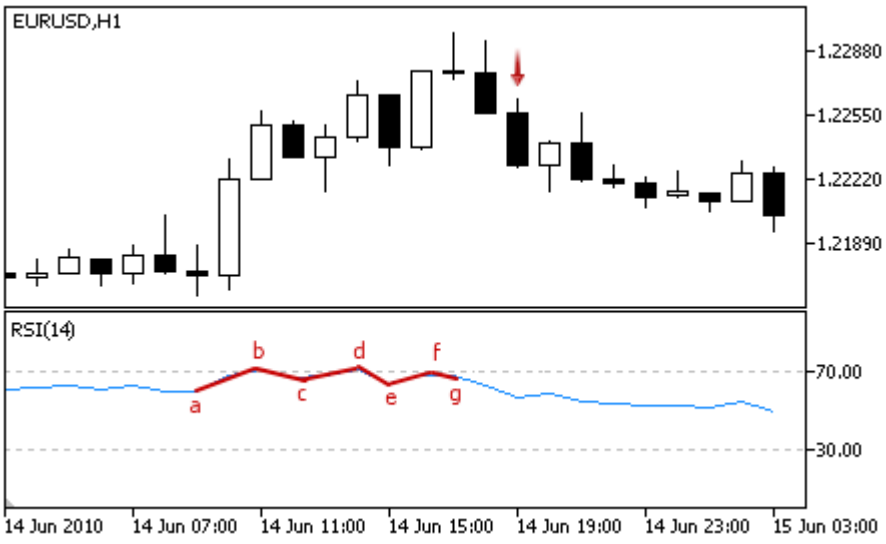
© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• 초과매수 수준 아래의 리버스 — 오실레이터가 하향으로 전환되고 분석된 막대의 값이 초과매수 수준(기본값 70) 뒤에 있습니다.  • 스윙 실패 — 오실레이터가 분석 막대의 이전 바닥보다 낮게 떨어집니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 확산 — 오실레이터의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다.  • 이중 확산 — 진동자는 세 개의 피크를 형성했고, 각각은 이전 피크보다 낮습니다. 그리고 가격은 세 개의 상응하는 피크를 형성했으며, 각각은 이전 피크보다 높습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • **Head/Shoulders** —오실레이터는 세 개의 피크를 형성하고 중간 피크가 다른 피크보다 높습니다. 
매수하는 데에 이의 없음	오실레이터의 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	오실레이터 값은 분석된 막대에서 떨어집니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodRSI	오실레이터의 계산 주기.
Applied	오실레이터 계산에 사용되는 가격 시리즈입니다.

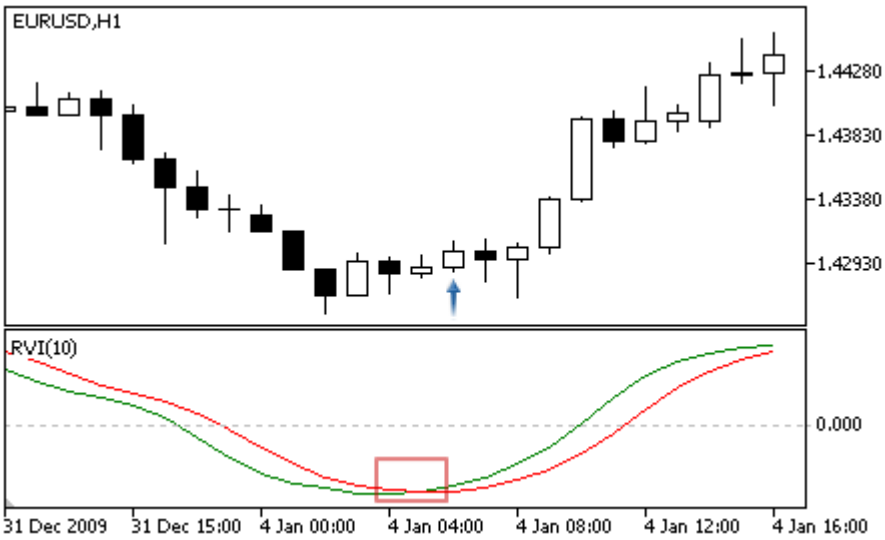
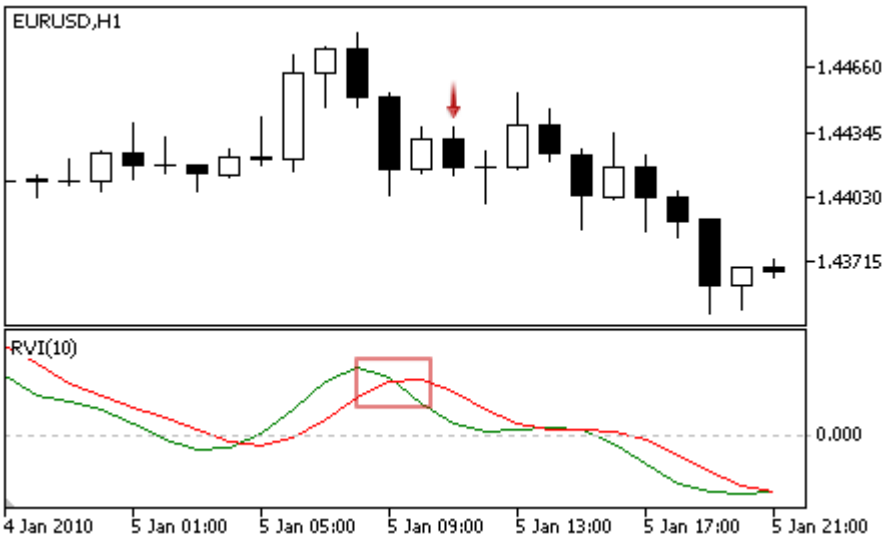
© 2000-2025, MetaQuotes Ltd.

오실레이터 상대적 활력 지수 시그널

이 시그널 모듈은 오실레이터 상대 활력 지수의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	**메인 및 시그널 라인 교차** —메인 선이 분석된 막대의 시그널 라인 위 및 이전 시그널 라인 아래에 있습니다. 
매도용	**메인 및 시그널 라인 교차** —메인 라인이 분석된 막대의 시그널 라인 아래 및 이전 막대의 시그널 라인 위에 있습니다. 
매수하는 데에 이의 없음	오실레이터의 값은 분석된 막대에서 증가합니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
매도하는 데에 이의 없음	오실레이터 값은 분석된 막대에서 떨어집니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodRVI	오실레이터의 계산 주기.

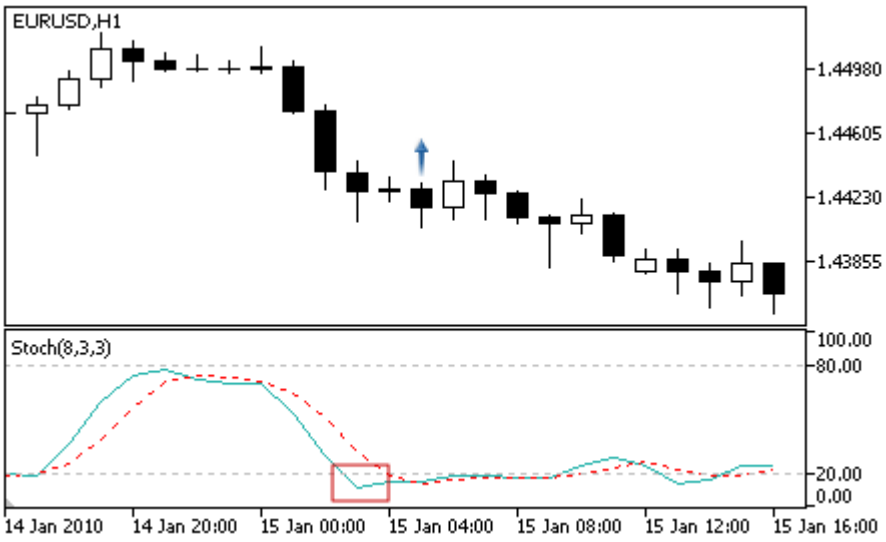
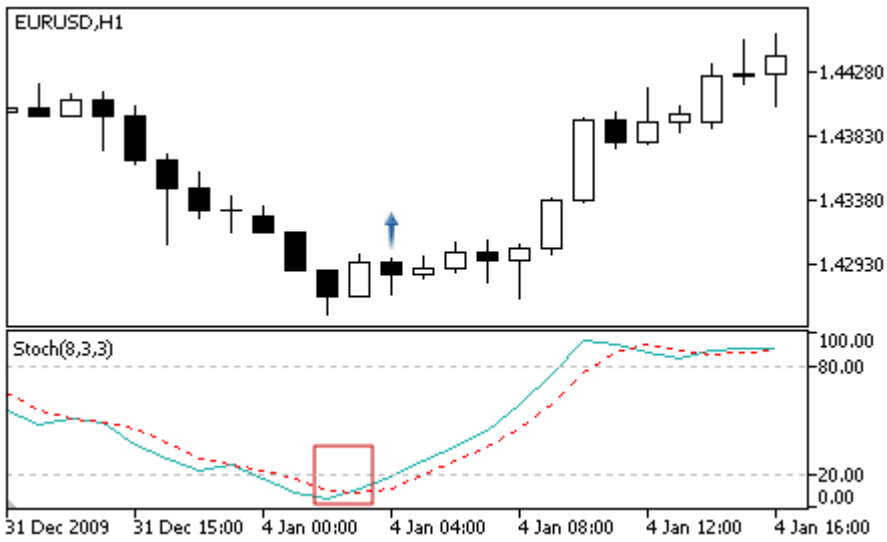
© 2000-2025, MetaQuotes Ltd.

확률적 오실레이터 시그널

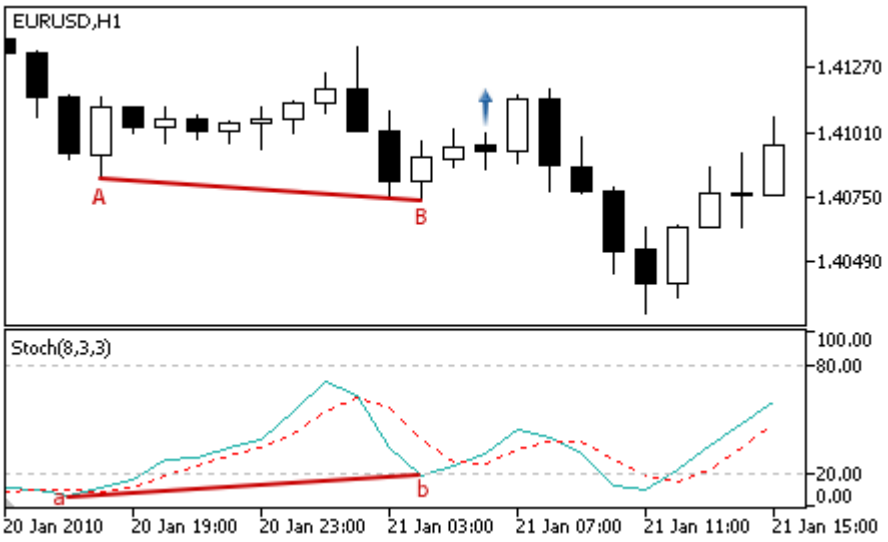
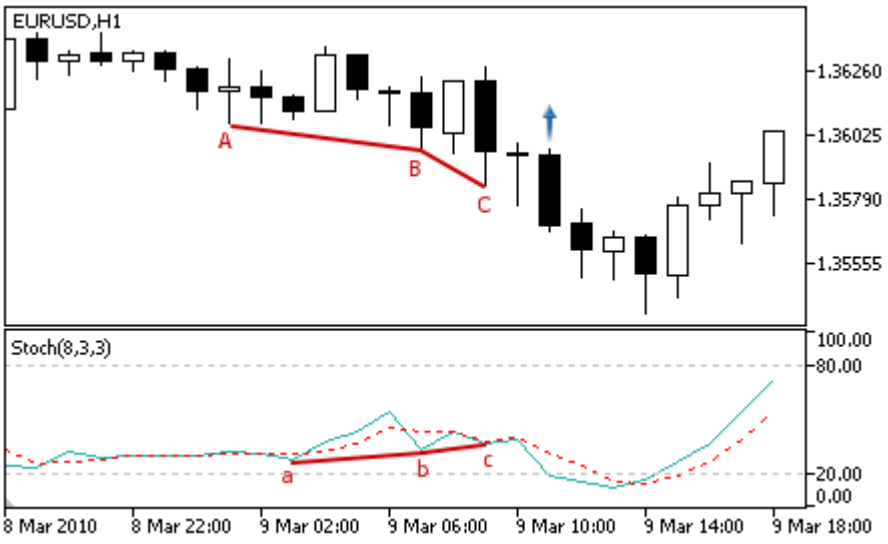
이 시그널 모듈은 오실레이터 Stochastic의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

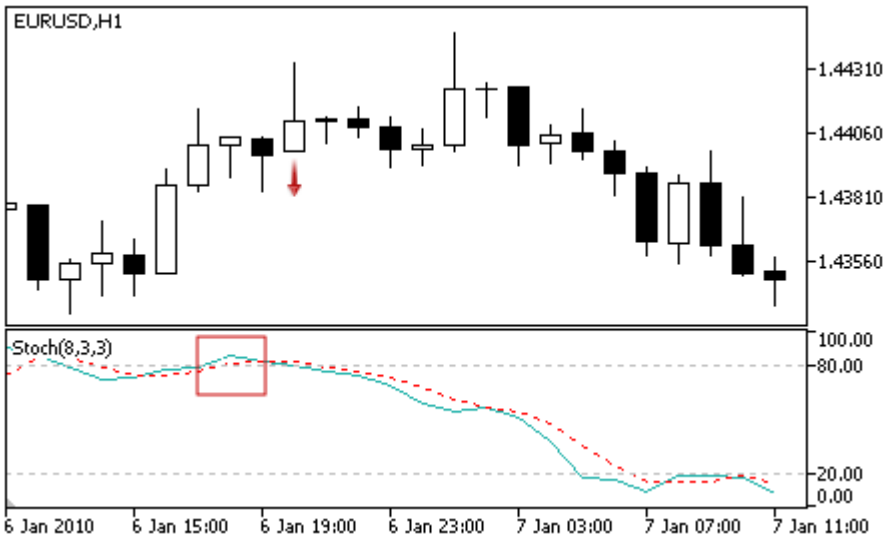
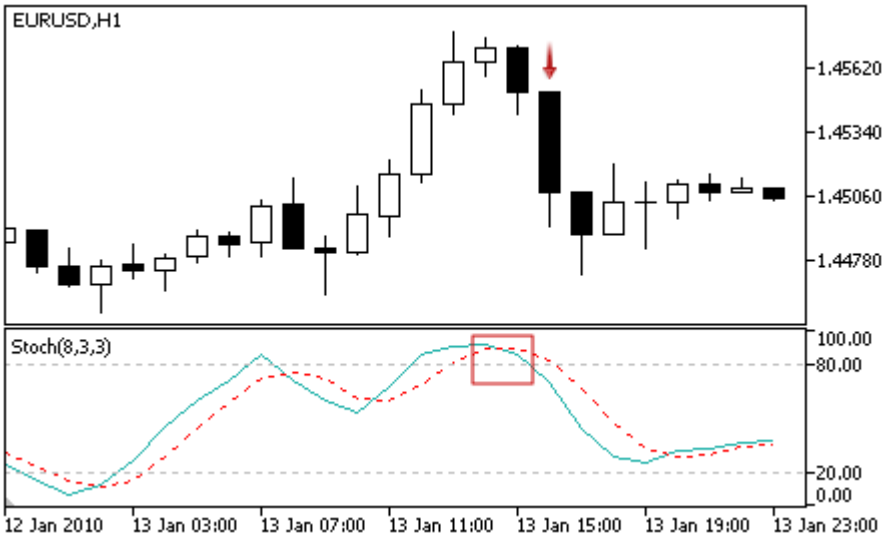
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• 리버스 —오실레이터가 위쪽으로 회전했음(오실레이터는 분석된 막대에서 상승하고 이전 막대에서 하락함).  • 메인 및 시그널 라인 교차 —메인 선이 분석된 막대의 시그널 라인 위 및 이전 시그널 라인 아래에 있습니다.  • 확산 —첫 번째 분석된 오실레이터의 바닥은 이전 것보다 높고 해당 가격 바닥은 이전 것보다 낮습니다.

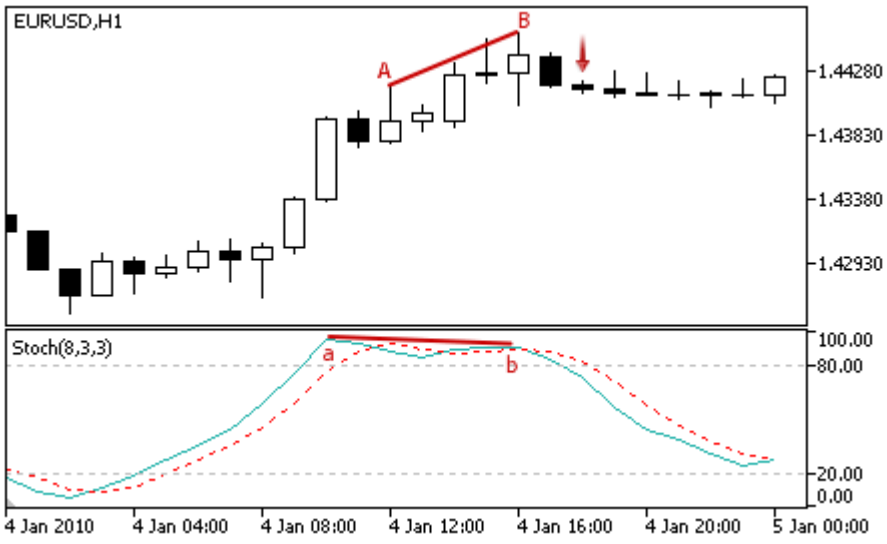
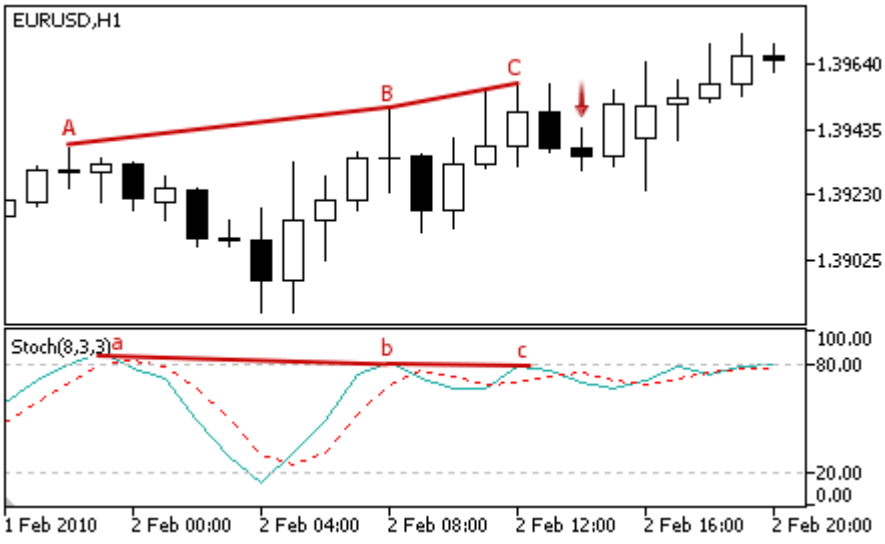
© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 이중 확산 —오실레이터는 3개의 결과 바닥을 형성하고, 각각은 이전 바닥을 형성하고, 가격은 3개의 상응하는 바닥을 형성하고, 각각은 이전 것보다 낮습니다. 
매도용	• 리버스 —오실레이터가 아래쪽으로 향했습니다(오실레이터는 분석된 막대에 떨어지며 이전 막대에 상승합니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 메인 및 시그널 라인 교차 —메인 라인이 분석된 막대의 시그널 라인 아래 및 이전 막대의 시그널 라인 위에 있습니다.  • 확산 —오실레이터의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 이중 확산 —진동자는 세 개의 피크를 형성했고, 각각은 이전 피크보다 낮습니다. 그리고 가격은 세 개의 상응하는 피크를 형성했으며, 각각은 이전 피크보다 높습니다. 
매수하는 데에 이의 없음	오실레이터의 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	오실레이터 값은 분석된 막대에서 떨어집니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodK	오실레이터의 메인 라인 계산 주기.
PeriodD	오실레이터의 주 라인 평균화 주기.
PeriodSlow	감속기간.
Applied	오실레이터 계산에 사용되는 가격 시리즈입니다.

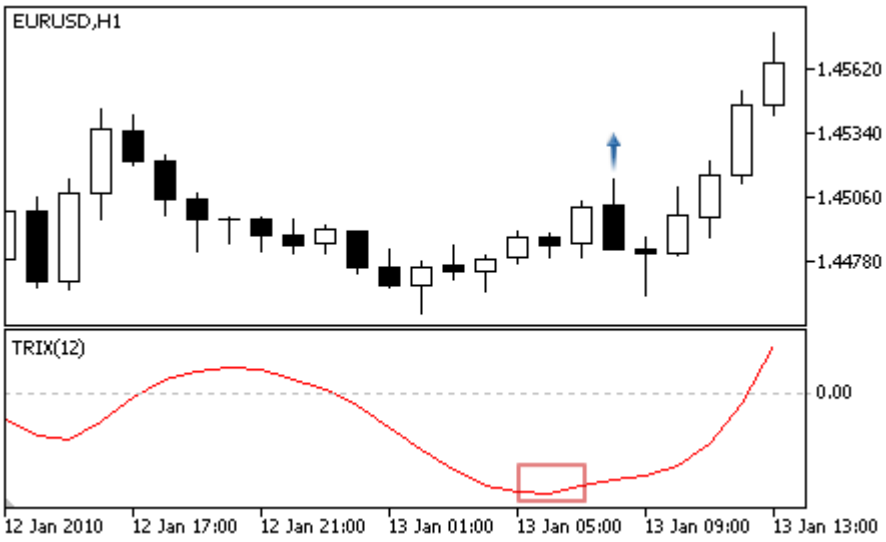
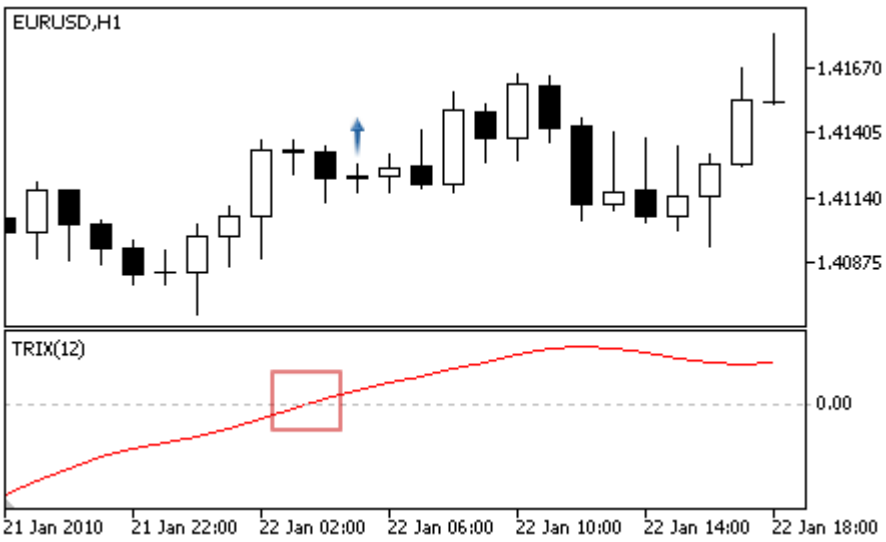
© 2000-2025, MetaQuotes Ltd.

오실레이터 삼중 지수 평균 시그널

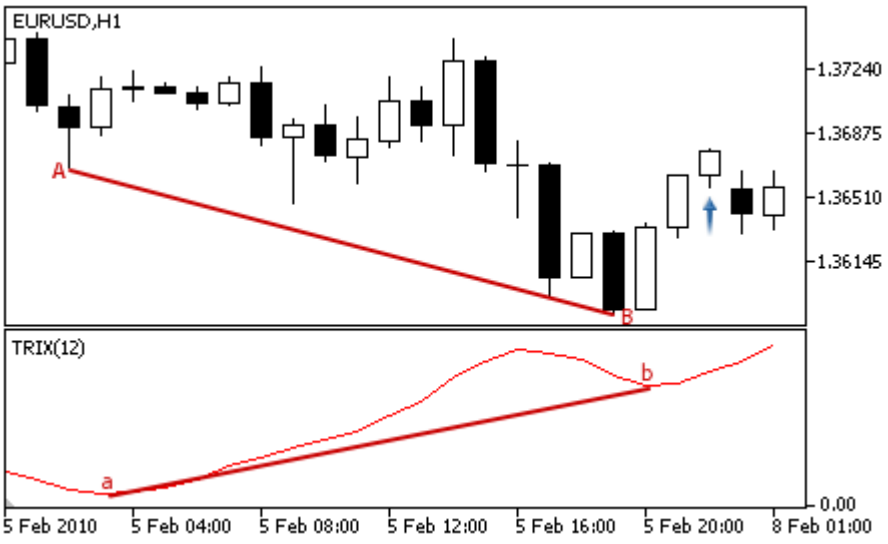
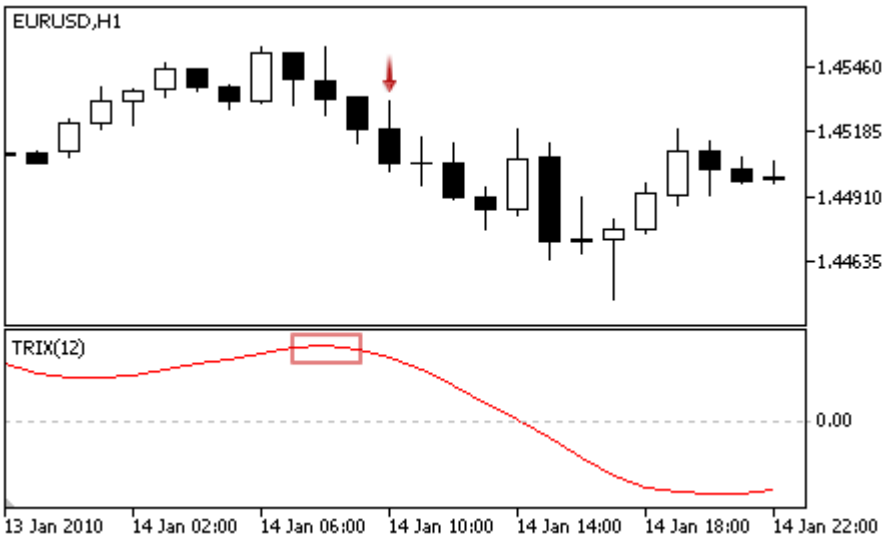
이 시그널 모듈은 오실레이터 삼중 지수 평균의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

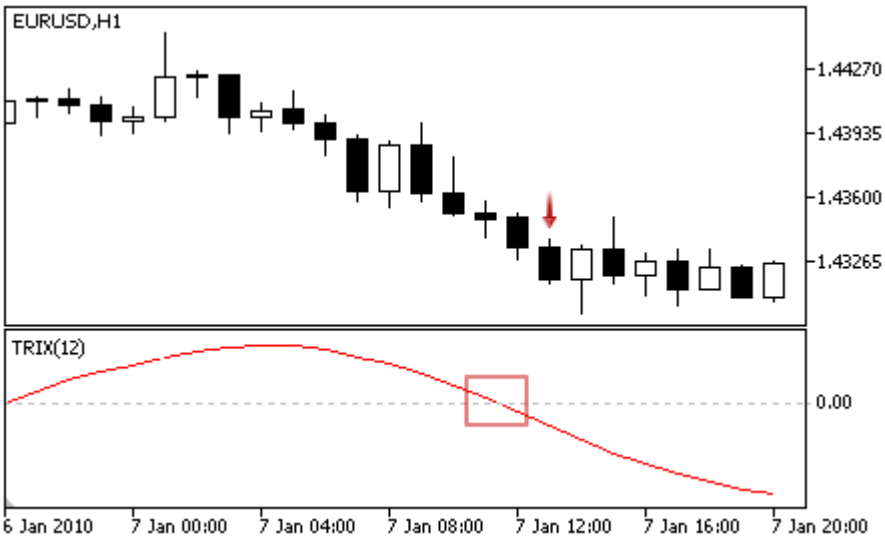
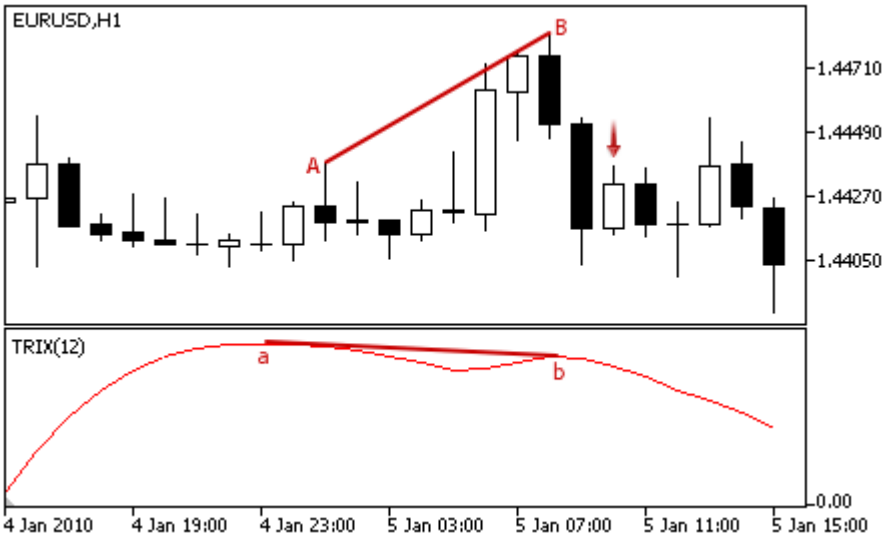
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **리버스** —오실레이터가 위쪽으로 회전했음(오실레이터는 분석된 막대에서 상승하고 이전 막대에서 하락함).  • **영점 교차** —메인 라인은 분석된 바에서는 0 레벨 위에 있으며 이전 바에서는 0 레벨 아래에 있습니다.  • **확산** —첫 번째 분석된 오실레이터의 바닥은 이전 것보다 높고 해당 가격 바닥은 이전 것보다 낮습니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• 리버스 —오실레이터가 아래쪽으로 향했습니다(오실레이터는 분석된 막대에 떨어지며 이전 막대에 상승합니다.  • 영점 교차 —메인 라인은 분석된 막대에서는 영점 이하이고 이전 바에서는 영점 이상입니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 확산 — 오실레이터의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다. 
매수하는 데에 이의 없음	오실레이터의 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	오실레이터 값은 분석된 막대에서 떨어집니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodTriX	오실레이터의 계산 주기.
Applied	오실레이터 계산에 사용되는 가격 시리즈입니다.

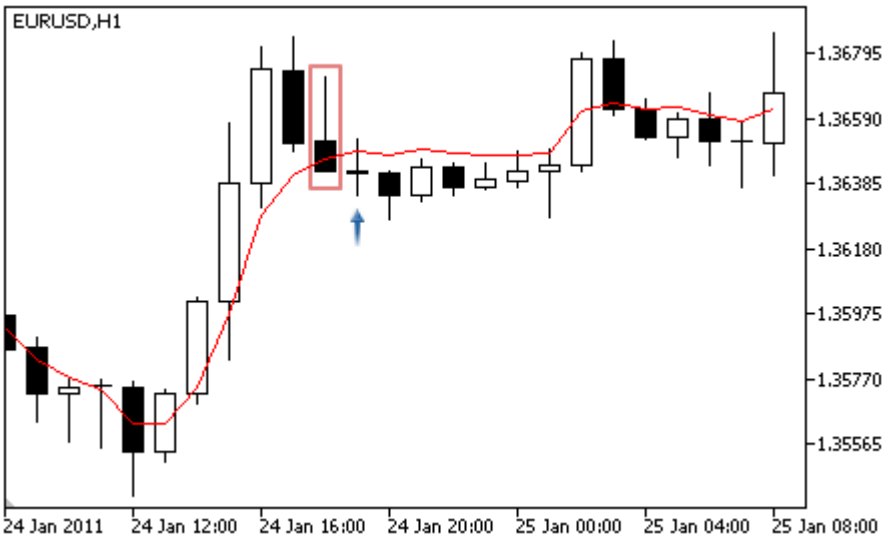
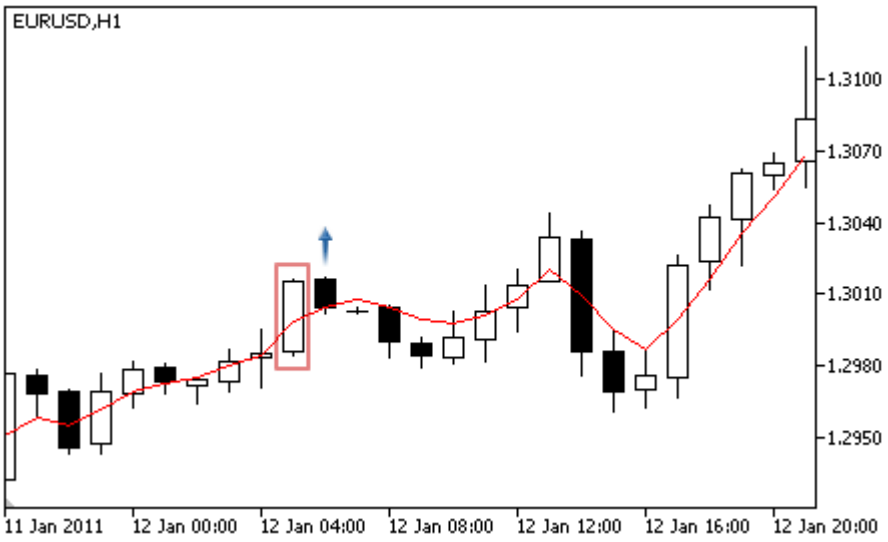
© 2000-2025, MetaQuotes Ltd.

삼중 지수 이동 평균 지표 시그널

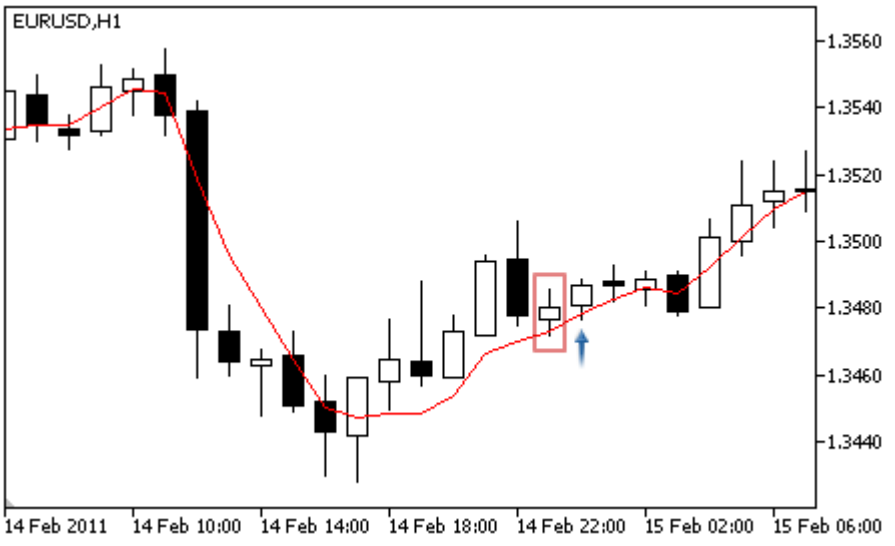
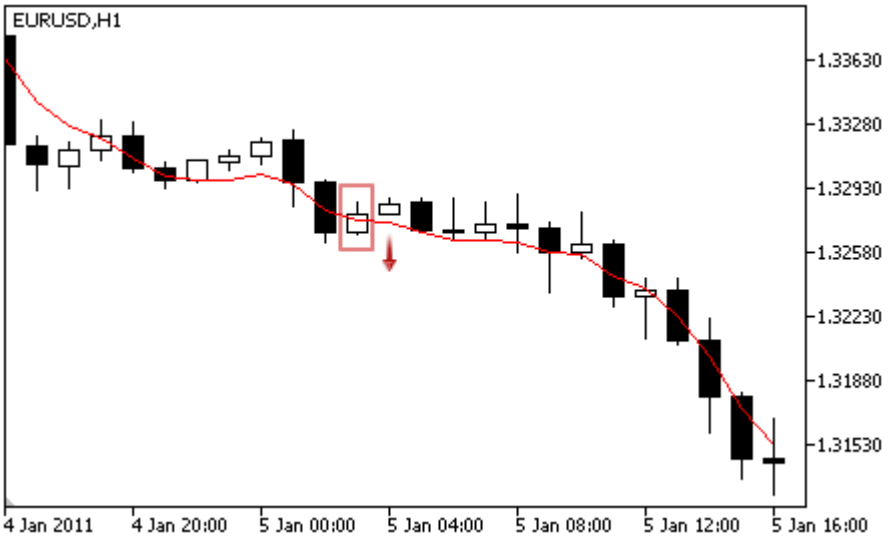
이 시그널 모듈은 삼중 지수 이동 평균 지표의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

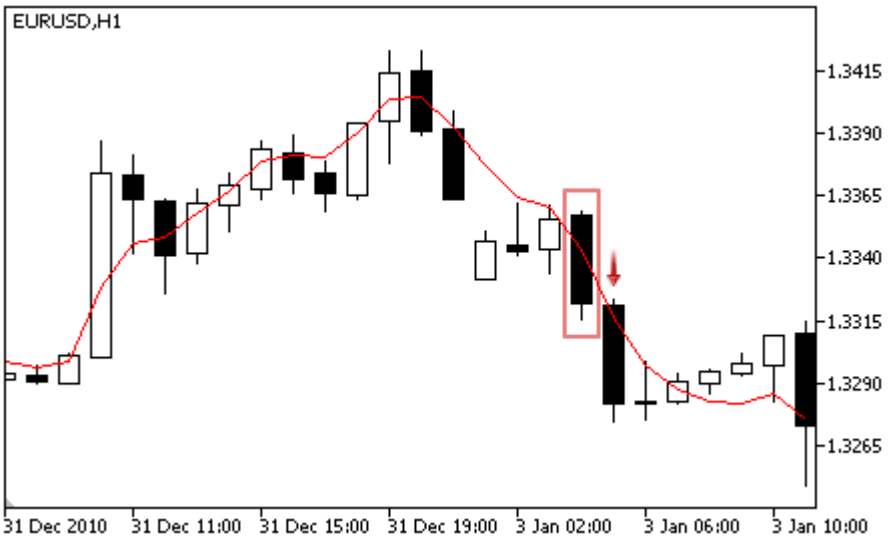
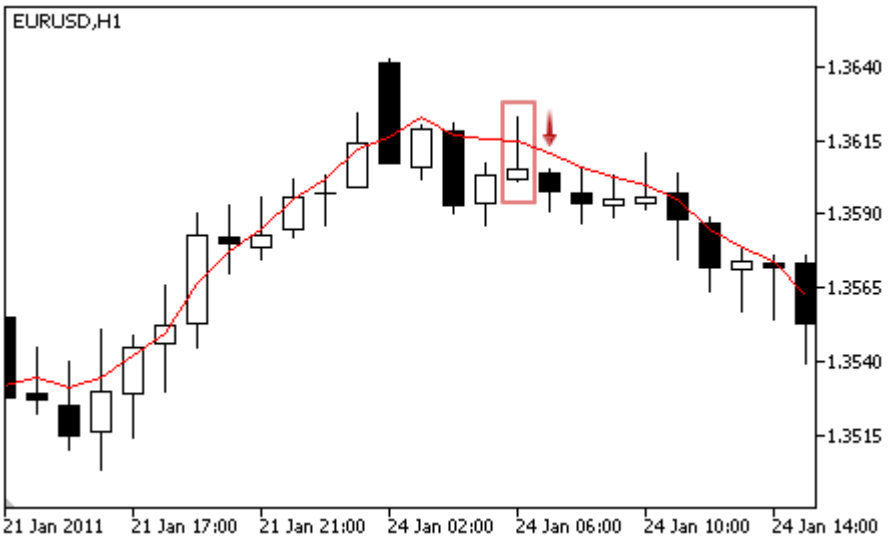
아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• 가격이 지표를 하향(분석된 막대의 시가는 지표보다 높고 종가는 지표보다 낮다)하고 지표가 상승(지표 선으로부터 롤백의 약세 신호)합니다.  • **이동 평균 교차**. 가격이 지표를 위로 통과했습니다(분석된 막대의 시가는 지표보다 낮고 종가는 지표보다 높습니다). 지표가 상승합니다(강세 시그널).  • 브레이크아웃 형성. 막대의 아래쪽 그림자가 지표(분석된 막대의 시가 및 종가는 지표 위에 있고 저가는 지표 아래에 있음)를 통과하고 지표가 상승합니다(지표 라인에서 롤백의 약세 신호).

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	
매도용	• **브레이크아웃 실패**. 가격이 지표 위로 교차했고(분석된 막대의 시가는 지표보다 낮고 종가는 지표보다 높음) 지표가 하락합니다(지표 라인으로부터 롤백의 약세 신호).  • **이동 평균 교차**. 가격이 지표를 하향(분석된 막대의 시가는 지표보다 높고 종가는 지표보다 낮음)하고 지표는 하락(강세 시그널)합니다.

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • **Formed breakout.** The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar are below the indicator, and the High price is above the indicator) and the indicator falls (weak signal of a roll-back from the indicator line). 
매수하는 데에 이의 없음	가격이 지표보다 위에 있습니다.
매도하는 데에 이의 없음	가격이 지표보다 아래에 있습니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodMA	지표의 평균화 기간.
Shift	시간 축을 따라 지표를 이동합니다(막대).
Method	평균화 메서드.
Applied	지표 계산에 사용되는 가격 시리즈.

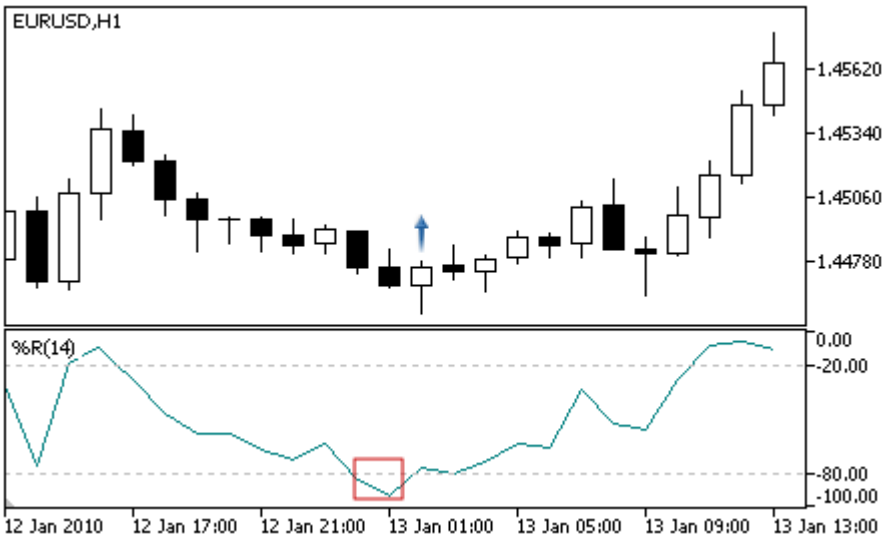
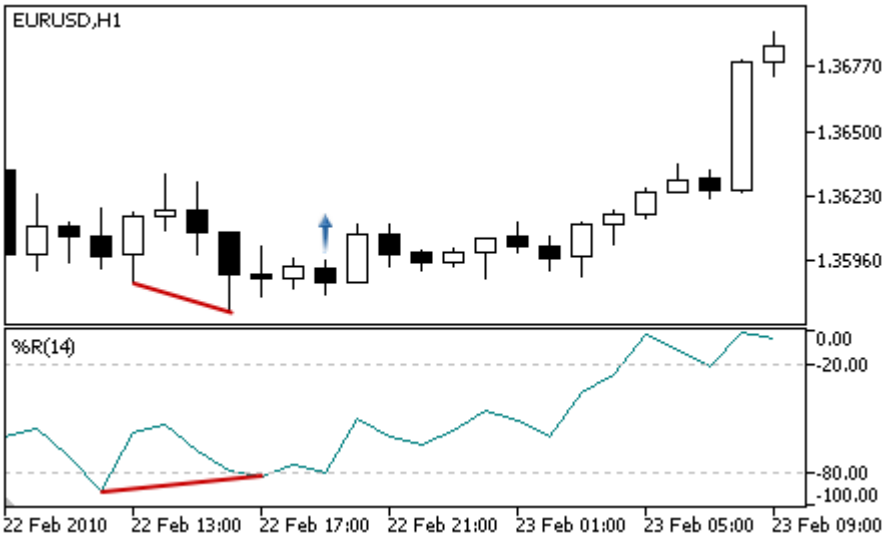
© 2000-2025, MetaQuotes Ltd.

오실레이터 윌리엄스 백분율 범위 시그널

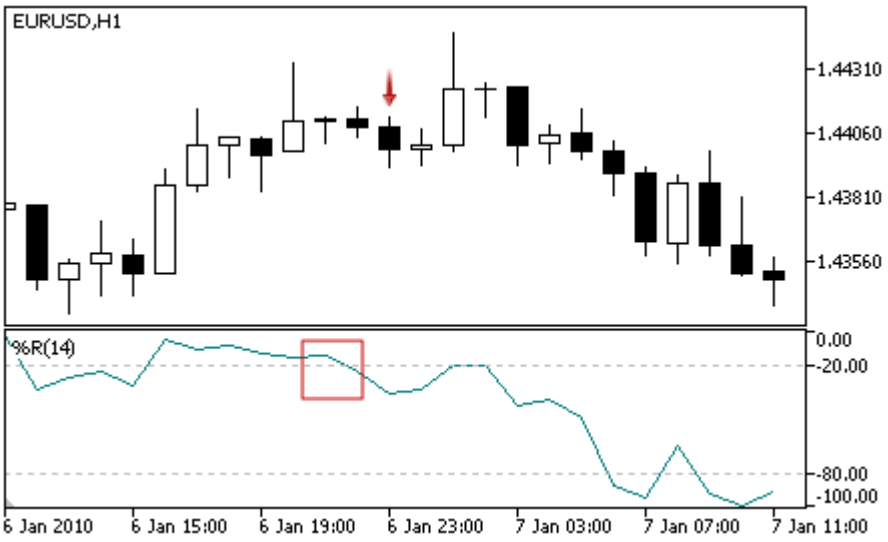
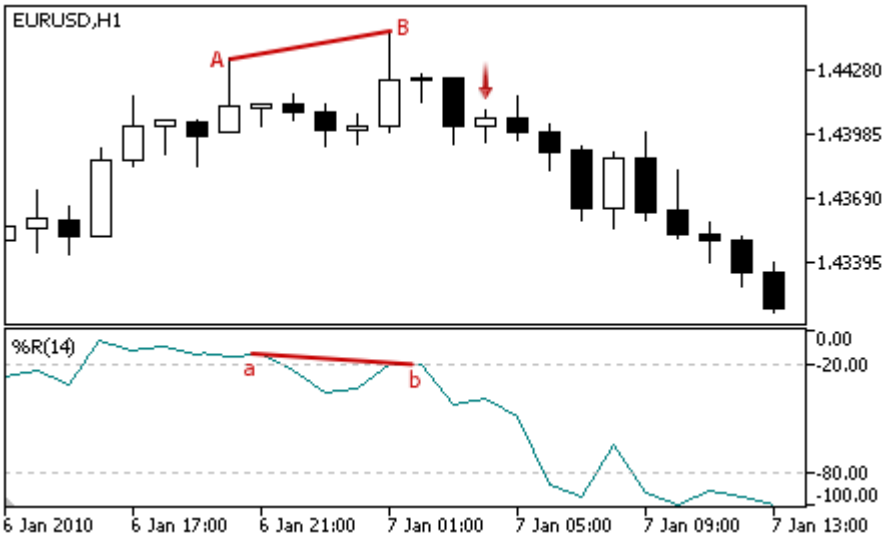
이 신호 모듈은 오실레이터 윌리엄스 백분율 범위의 시장 모델을 기반으로 합니다. 모듈로부터 얻은 신호에 기초하여 무역 결정을 내리는 메커니즘은 별도 섹션에 설명되어 있습니다.

시그널 발생 조건

아래에서 모듈이 Expert Advisor에게 시그널을 전달할 때의 조건에 대한 설명을 볼 수 있습니다.

시그널 유형	조건 설명
매수	• **초과판매 수준 아래의 리버스** — 오실레이터가 위쪽으로 회전하고 분석된 바의 값이 초과 판매 수준(기본값 -80)에 미치지 않습니다.  • **확산** — 첫 번째 분석된 오실레이터의 바닥은 이전 것보다 높고 해당 가격 바닥은 이전 것보다 낮습니다. 
매도용	• **초과매수 수준 아래의 리버스** — 오실레이터가 하향으로 전환되고 분석된 바의 값이 초과매수 수준 아래에 있음(기본값은 -20임).

© 2000-2025, MetaQuotes Ltd.

시그널 유형	조건 설명
	 • 확산 — 오실레이터의 첫 번째 분석 피크는 이전 피크보다 낮고 해당 가격 피크는 이전 피크보다 높습니다. 
매수하는 데에 이의 없음	오실레이터의 값은 분석된 막대에서 증가합니다.
매도하는 데에 이의 없음	오실레이터 값은 분석된 막대에서 떨어집니다.

참고

Expert Advisor의 작동 모드에 따라("모든 틱" 또는 "시가 전용") 분석 막대는 현재 막대(색인 0) 또는 마지막으로 형성된 막대(인덱스 1)입니다.

오실레이터 윌리엄스 백분율 범위의 척도는 반대로 표시됩니다. 최대값은 -100이고 최소값은 0입니다.

조정 가능한 매개 변수

이 모듈에는 다음과 같은 조정 가능한 매개 변수가 있습니다:

© 2000-2025, MetaQuotes Ltd.

Parameter	Description
Weight	0에서 1 사이의 모듈 신호 무게입니다.
PeriodWPR	오실레이터의 계산 주기.

© 2000-2025, MetaQuotes Ltd.

추적 손절매 클래스

이 절에는 추적 손절매 클래스 관련 작업에 대한 기술 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다.

이러한 클래스를 사용하면 거래 전략을 만들고 테스트할 때 시간을 절약할 수 있습니다.

MQL5 표준 라이브러리(거래 전략 측면)는 터미널 디렉토리의 Include₩Expert₩Trailing 폴더에 배치됩니다.

Class	Description
CTrailingFixedPips	이 클래스는 고정점을 기반으로 추적 손절매 알고리즘을 구현합니다
CTrailingMA	이 클래스는 이동 평균 지표 값을 기반으로 추적 손절매 알고리즘을 구현합니다
CTrailingNone	스텁 클래스, 추적 손절매 알고리즘을 사용하지 않음
CTrailingPSAR	이 클래스는 파라볼릭 SAR 지표 값을 기반으로 추적 손절매 알고리즘을 구현합니다

© 2000-2025, MetaQuotes Ltd.

CTrailingFixedPips

CTrailingFixedPips는 고정 지점 추적에 기반한 추적 손절매 알고리즘이 구현된 클래스입니다.

CTrailingFixedPips는 고정 지점 추적에 기반한 추적 손절매 알고리즘이 구현된 클래스입니다. 그렇지 않으면, 가격이 유리한 방향으로 이동했는지 여부를 확인하는 작업이 수행됩니다.

포지션에 손실 중지 명령이 있는 경우 현재 가격 대비 최소 허용 손실 중지 차이를 확인합니다. 포지션에 손절매 수준이 없는 경우 현재 가격과 시가 사이의 차이가 확인됩니다. 거리가 손절매 수준을 초과하면 시스템에서 새로운 손절매 가격을 설정할 것을 시사합니다.

손절매 수정 조건이 충족되고 이익실현이 0이 아닌 경우, 시스템은 새로운 이익실현 가격을 설정할 것을 제안합니다.

Expert Advisor 클래스가 every_tick=false 플래그로 Initialize된 경우, 작업 기호 및 시간 상의 새 표시줄에서만 모든 연산(후행 등)을 수행합니다. 이 경우 이익 실현 주문을 설정하면 새 막대가 나타나기 전에 가격이 포지션 방향으로 이동할 때 포지션을 클로즈 할 수 있습니다.

Description

CTrailingFixedPips는 현재 가격(포인트 단위)에서 지정된 "차이"에서 추적 손절매 알고리즘을 구현합니다.

Declaration

```
class CTrailingFixedPips: public CExpertTrailing
```

Title

```
   #include <Expert\Trailing\CTrailingFixedPips.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertTrailing
      CTrailingFixedPips

그룹별 클래스 메서드

Initialization	
StopLevel	손실 정지 수준의 값 설정
ProfitLevel	이익 실현 수준의 값 설정
virtual ValidationSettings	설정 확인
추적 메서드 확인	
virtual CheckTrailingStopLong	롱 포지션의 추적 손절매 상태 점검

© 2000-2025, MetaQuotes Ltd.

Initialization	
virtual CheckTrailingStopShort	숏 포지션의 추적 손절매 상태 점검

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

© 2000-2025, MetaQuotes Ltd.

StopLevel

손절매 수준의 값을 설정합니다.

```
void  StopLevel(
   int  stop_level      // 수준
   )
```

Parameters

stop_loss

[in] 손절매 수준 값(기존 2/4자리 점).

참고

손절매 수준이 0이면 추적 손절매가 사용되지 않습니다.

© 2000-2025, MetaQuotes Ltd.

ProfitLevel

이익실현 수준 값을 설정합니다.

```
void  ProfitLevel(
   int  profit_level      // 수준
   )
```

Parameters

profit_level

[in] 이익 수준 획득 값(기존 2/4자리 포인트).

참고

이익실현 수준이 0이면 추적 손절매가 사용되지 않습니다.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

성공이면 true, 그렇지 않으면 false.

참고

이 기능은 이익실현 및 추적 손절매 수준을 확인합니다. 올바른 값은 0이며 현재 종가에서 스탑 주문까지의 점 표시 최소값보다 큰 값입니다.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

롱 포지션의 추적 손절매 조건을 점검합니다.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 손절매 참조
   double&         tp            // 이익실현 참조
   )
```

Parameters

position

[in] CPositionInfo 객체에 대한 포인터.

sl

[in][out] 손절매 가격 변수 참조.

tp

[in][out] 수익률 획득에 대한 변수 참조.

반환 값

조건이 만족되면 -true, 그렇지 않으면 -false.

참고

손절매 수준이 0이면 조건이 충족되지 않습니다(종료). 포지션에 이미 손절매 가격이 있는 경우, 그 값은 기준가격으로 가정되며, 그렇지 않은 경우 포지션 시가는 기준 가격으로 가정됩니다. 현재 입찰 가격이 기준 가격보다 높으면 손절매 가격을 새로 설정할 것을 제안합니다. 이 경우 포지션에 이미 이익실현이 있다면 신규 이익실현 가격을 입찰가+이익 실현 수준으로 책정하는 것이 좋다.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

숏 포지션의 추적 손절매 상태를 점검합니다.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 손절매 참조
   double&         tp            // 이익실현 참조
   )
```

Parameters

position

[in] CPositionInfo 객체에 대한 포인터.

sl

[in][out] 손절매 가격 변수 참조.

tp

[in][out] 수익률 획득에 대한 변수 참조.

반환 값

조건이 만족되면 -true, 그렇지 않으면 -false.

참고

손절매 수준이 0이면 조건이 충족되지 않습니다(종료). 포지션에 이미 손절매 가격이 있는 경우, 그 값은 기준가격으로 가정되며, 그렇지 않은 경우 포지션 시가는 기준 가격으로 가정됩니다. 현재 호가가 기준 가격 -손절매 수준보다 낮으면 새로운 손절매 가격을 설정하는 것이 좋습니다. 이 경우, 포지션에 이미 이익실현이 있다면 새로운 이익실현 가격을 호가 -이익실현 수준으로 설정하는 것이 좋습니다.

© 2000-2025, MetaQuotes Ltd.

CTrailingMA

CTrailingMA는 이동 평균 지표 값을 기반으로 추적 손절매 알고리즘을 구현하는 클래스입니다.

Description

CTrailingMA 클래스 추적 손절매 알고리즘은 이전의(완료)바의 평균 지표가 움직이는 것을 값을 바탕으로 구현합니다.

Declaration

```
class CTrailingMA: public CExpertTrailing
```

Title

```
   #include <Expert\Trailing\TrailingMA.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertTrailing
      CTrailingMA

그룹별 클래스 메서드

Initialization	
Period	이동 평균 기간 설정
Shift	이동 평균 시프트 설정
Method	이동 평균의 평활 방법을 설정합니다
Applied	이동 평균 적용 가격 설정
virtual InitIndicators	지표 및 시간대 초기화
virtual ValidationSettings	설정 확인
추적 메서드 확인	
virtual CheckTrailingStopLong	롱 포지션의 추적 손절매 상태 점검
virtual CheckTrailingStopShort	숏 포지션의 추적 손절매 상태 점검

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

Period

이동 평균의 주기를 설정합니다.

```
void  Period(
   int  period      // 기간
   )
```

Parameters

period

[in] 이동평균의 기간.

© 2000-2025, MetaQuotes Ltd.

Shift

이동 평균의 시프트를 설정합니다.

```
void  Shift(
   int  shift      // 파라미터
   )
```

Parameters

shift

[in] 이동 평균의 시프트.

© 2000-2025, MetaQuotes Ltd.

Method

이동 평균의 평활 방법을 설정합니다.

```
void  Method(
   ENUM_MA_METHOD  method      // 파라미터
   )
```

Parameters

method

[in] 이동 평균의 평활 메서드 (ENUM_MA_METHOD 열거값으로부터).

© 2000-2025, MetaQuotes Ltd.

Applied

이동 평균의 적용 가격을 설정합니다.

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // 파라미터
   )
```

Parameters

applied

[in] 이동 평균 적용 가격 (ENUM_APPLIED_PRICE 열거값으로부터).

© 2000-2025, MetaQuotes Ltd.

InitIndicators

지표 및 시간대를 초기화합니다.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터 (CExpert 클래스 멤버).

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

성공이면 true, 그렇지 않으면 false.

참고

이 함수는 평균 이동 주기를 점검하며, 정확한 값은 양수입니다.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

롱 포지션의 추적 손절매 조건을 점검합니다.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 참조
   double&         tp            // 참조
   )
```

Parameters

position

[in] CPositionInfo 객체에 대한 포인터.

sl

[in][out] 손절매 가격 변수 참조.

tp

[in][out] 수익률 획득에 대한 변수 참조.

반환 값

조건이 만족되면 -true, 그렇지 않으면 -false.

참고

먼저 현재 가격에 가장 가까운 최대 허용 손절매 가격을 계산하고 이전(완료) 막대의 이동 평균 지표 값을 사용하여 손절매 가격을 계산합니다. 포지션에 이미 손절매 가격이 있는 경우, 그 값은 기준가격으로 가정되며, 그렇지 않은 경우 기준가격은 포지션의 시가입니다. 계산된 손절매 가격이 기준 가격보다 높고 최대 허용 손절매 가격보다 낮으면 새로운 손절매 가격을 설정할 것을 제안합니다.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

숏 포지션의 추적 손절매 상태를 점검합니다.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 참조
   double&         tp            // 참조
   )
```

Parameters

position

[in] CPositionInfo 객체에 대한 포인터.

sl

[in][out] 손절매 가격 변수 참조.

tp

[in][out] 수익률 획득에 대한 변수 참조.

반환 값

조건이 만족되면 -true, 그렇지 않으면 -false.

참고

먼저 현재 가격에 가장 가까운 최소 허용 손절매 가격을 계산하고 이전(완료) 막대의 이동 평균 지표 값을 사용하여 손절매 가격을 계산합니다. 포지션에 이미 손절매 가격이 있는 경우, 그 값은 기준가격으로 가정되며, 그렇지 않은 경우 기준가격은 포지션의 시가입니다. 계산된 손절매 가격이 기준 가격보다 낮고 최소 허용 손절매 가격보다 높으면 손절매 가격을 새로 설정하는 것이 좋습니다.

© 2000-2025, MetaQuotes Ltd.

CTrailingNone

CTrailingNone은 stub 클래스입니다. 전략에서 추적 손절매를 사용하지 않는 경우 CExpert 클래스의 추적 객체를 초기화할 때 이 클래스를 사용해야 합니다.

Description

CTrailingNone 클래스는 추적 손절매 알고리즘을 구현하지 않습니다. 추적 손절매 조건을 확인하는 방법은 항상 false를 반환한다.

Declaration

```
class CTrailingNone: public CExpertTrailing
```

Title

```
   #include <Expert\Trailing\TrailingNone.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertTrailing
      CTrailingNone

그룹별 클래스 메서드

추적 메서드 확인	
virtual CheckTrailingStopLong	롱 포지션의 추적 손절매 조건을 점검하는 stub 방법
virtual CheckTrailingStopShort	숏 포지션의 추적 손절매 상태를 점검하는 stub 방법

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

CExpertTrailing 클래스에서 상속된 방법

CheckTrailingStopLong, CheckTrailingStopShort

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

롱 포지션의 추적 손절매 조건을 점검합니다.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 참조
   double&         tp            // 참조
   )
```

Parameters

position

[in] CPositionInfo 객체에 대한 포인터.

sl

[in][out] 손절매 가격 변수 참조.

tp

[in][out] 수익률 획득에 대한 변수 참조.

반환 값

조건이 만족되면 -true, 그렇지 않으면 -false.

참고

함수는 항상 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

숏 포지션의 추적 손절매 상태를 점검합니다.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 참조
   double&         tp            // 참조
   )
```

Parameters

position

[in] CPositionInfo 객체에 대한 포인터.

sl

[in][out] 손절매 가격 변수 참조.

tp

[in][out] 수익률 획득에 대한 변수 참조.

반환 값

조건이 만족되면 -true, 그렇지 않으면 -false.

참고

함수는 항상 false를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CTrailingPSAR

CTrailingPSAR은 파라볼릭 SAR 표시기 값을 기반으로 추적 손절매 알고리즘을 구현하는 클래스입니다.

Description

CTrailingPSAR 클래스는 이전(완료된) 막대의 파라볼릭 SAR 지표 값을 기반으로 추적 손절매 알고리즘을 구현합니다.

Declaration

```
class CTrailingPSAR: public CExpertTrailing
```

Title

```
#include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertTrailing
      CTrailingPSAR

그룹별 클래스 메서드

Initialization	
Step	파라볼릭 SAR 지표의 단계 값을 설정합니다.
Maximum	파라볼릭 SAR 지표의 최대값 설정
virtual InitIndicators	지표 및 시간대 초기화
추적 메서드 확인	
virtual CheckTrailingStopLong	롱 포지션의 추적 손절매 상태 점검
virtual CheckTrailingStopShort	숏 포지션의 추적 손절매 상태 점검

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries

© 2000-2025, MetaQuotes Ltd.

Step

파라볼릭 SAR 지표의 단계 값을 설정합니다.

```
void  Step(
   double  step      // 파라미터
   )
```

Parameters

step

[in] 파라볼릭 SAR 지표의 단계 값.

© 2000-2025, MetaQuotes Ltd.

Maximum

파라볼릭 SAR 지표의 최대값을 설정합니다.

```
void  Maximum(
   double  maximum      // 파라미터
   )
```

Parameters

maximum

[in] 파라볼릭 SAR 지표의 최대값.

© 2000-2025, MetaQuotes Ltd.

InitIndicators

지표 및 시간대를 초기화합니다.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // 포인터
   )
```

Parameters

indicators

[in] 지표 및 시계열 모음에 대한 포인터 (CExpert 클래스 멤버).

반환 값

성공이면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopLong

롱 포지션의 추적 손절매 조건을 점검합니다.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 참조
   double&         tp            // 참조
   )
```

Parameters

position

[in] CPositionInfo 객체에 대한 포인터.

sl

[in][out] 손절매 가격 변수 참조.

tp

[in][out] 수익률 획득에 대한 변수 참조.

반환 값

조건이 만족되면 -true, 그렇지 않으면 -false.

참고

먼저 현재 가격에 가장 가까운 최대 허용 손절매 가격을 계산하고 이전(완료된) 막대의 파라볼릭 SAR 지표 값을 사용하여 손절매 가격을 계산합니다. 포지션에 이미 손절매 가격이 있는 경우, 그 값은 기준 가격으로 가정되며, 그렇지 않은 경우 포지션 시가는 기준 가격으로 가정됩니다. 계산된 손절매 가격이 기준 가격보다 높고 최대 허용 손절매 가격보다 낮으면 새로운 손절매 가격을 설정할 것을 제안합니다.

© 2000-2025, MetaQuotes Ltd.

CheckTrailingStopShort

숏 포지션의 추적 손절매 상태를 점검합니다.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // 포인터
   double&         sl,           // 참조
   double&         tp            // 참조
   )
```

Parameters

position

[in] CPositionInfo 객체에 대한 포인터.

sl

[in][out] 손절매 가격 변수 참조.

tp

[in][out] 수익률 획득에 대한 변수 참조.

반환 값

조건이 만족되면 -true, 그렇지 않으면 -false.

참고

먼저 현재 가격에 가장 가까운 최소 허용 손절매 가격을 계산하고 이전(완료된) 막대의 파라볼릭 SAR 지표 값을 사용하여 손절매 가격을 계산합니다. 포지션에 이미 손절매 가격이 있는 경우, 그 값은 기준 가격으로 가정되며, 그렇지 않은 경우 포지션 시가는 기준 가격으로 가정됩니다. 계산된 손절매 가격이 기준 가격보다 낮고 최소 허용 손절매 가격보다 높으면 손절매 가격을 새로 설정하는 것이 좋습니다.

© 2000-2025, MetaQuotes Ltd.

자금 관리 클래스

이 절에는 비용 및 리스크 관리 클래스 작업에 대한 기술적 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다.

이러한 클래스를 사용하면 거래 전략을 만들고 테스트할 때 시간을 절약할 수 있습니다.

MQL5 표준 라이브러리(비용 및 리스크 관리 클래스 기준)는 터미널 디렉터리인 Include₩Expert₩Money₩에 배치됩니다.

Class	Description
CMoneyFixedLot	이 클래스는 미리 정의된 고정 로트 크기를 가진 거래를 기반으로 자금 관리 알고리즘을 구현합니다.
CMoneyFixedMargin	이 클래스는 미리 정의된 고정 마진을 가진 거래를 기반으로 자금 관리 알고리즘을 구현합니다.
CMoneyFixedRisk	이 클래스는 리스크가 미리 정의된 거래를 기반으로 자금 관리 알고리즘을 구현합니다.
CMoneyNone	이 클래스는 최소 허용 로트 크기로 거래를 기반으로 자금 관리 알고리즘을 구현합니다.
CMoneySizeOptimized	이 클래스는 이전 거래의 결과에 따라 변동 로트 크기의 거래를 기반으로 자금 관리 알고리즘을 구현합니다.

© 2000-2025, MetaQuotes Ltd.

CMoneyFixedLot

CMoneyFixedLot은 미리 정의된 고정 로트 크기를 가진 거래를 기반으로 자금 관리 알고리즘을 구현하도록 설계된 클래스입니다.

Description

CMoneyFixedLot은 미리 정의된 고정 로트 크기를 가진 거래를 기반으로 자금 관리 알고리즘을 구현합니다.

Declaration

```
class CMoneyFixedLot: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedLot

그룹별 클래스 메서드

Initialization	
Lots	거래량을 설정합니다
virtual ValidationSettings	설정 확인
자금 및 위험 관리 메서드	
virtual CheckOpenLong	롱 포지션에 대한 거래량을 가져옵니다
virtual CheckOpenShort	숏 포지션에 대한 거래량을 가져옵니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

CExpertMoney 클래스에서 상속된 방법

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

Lots

거래량을 설정합니다(로트 단위).

```
void  Lots(
   double  lots      // 로트 수
   )
```

Parameters

lots

[in] 거래량.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

성공이면 true, 그렇지 않으면 false.

참고

미리 정의된 고정 로트가 허용 값의 범위 내에 있고 변경 단계의 다중인지 확인합니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

롱 포지션에 대한 거래량을 가져옵니다.

```
virtual double  CheckOpenLong(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

롱 포지션의 거래량.

참고

이 함수는 항상 미리 정의된 고정 거래량을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

숏 포지션용 거래량을 가져옵니다.

```
virtual double  CheckOpenShort(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

숏 포지션용 거래량.

참고

이 함수는 항상 미리 정의된 고정 거래량을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CMoneyFixedMargin

CMoneyFixedMargin은 미리 정의된 고정 마진을 가진 거래를 기반으로 자금 관리 알고리즘을 구현하기 위해 고안된 클래스입니다.

Description

CMoneyFixedMargin은 미리 정의된 고정 마진을 가진 거래를 기반으로 자금 관리 알고리즘을 구현합니다 .

Declaration

```
class CMoneyFixedMargin: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedMargin

그룹별 클래스 메서드

자금 및 위험 관리 메서드	
virtual CheckOpenLong	롱 포지션에 대한 거래량을 가져옵니다
virtual CheckOpenShort	숏 포지션에 대한 거래량을 가져옵니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

CExpertMoney 클래스에서 상속된 방법

Percent, ValidationSettings, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

롱 포지션에 대한 거래량을 가져옵니다.

```
virtual double  CheckOpenLong(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

롱 포지션의 거래량.

참고

이 함수는 미리 정의된 마진이 고정된 롱 포지션에 대한 거래량을 반환합니다. CExpertMoney 기본 클래스의 백분율 매개 변수로 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

다음에 대한 거래량을 가져옵니다: 숏 포지션.

```
virtual double  CheckOpenShort(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

숏 포지션용 거래량.

참고

이 함수는 미리 정의된 고정 마진을 가진 숏 포지션에 대한 거래량을 반환합니다. CExpertMoney 기본 클래스의 백분율 매개 변수로 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

CMoneyFixedRisk

CMoneyFixedRisk는 미리 정의된 리스크가 고정된 자금 관리 알고리즘을 구현하는 클래스입니다.

Description

CMoneyFixedRisk 클래스는 미리 정의된 리스크가 고정된 자금 관리 알고리즘을 구현합니다.

Declaration

```
class CMoneyFixedRisk: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedRisk

그룹별 클래스 메서드

자금 및 위험 관리 메서드	
virtual CheckOpenLong	롱 포지션에 대한 거래량을 가져옵니다
virtual CheckOpenShort	숏 포지션에 대한 거래량을 가져옵니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

CExpertMoney 클래스에서 상속된 방법

Percent, ValidationSettings, CheckReverse

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

롱 포지션에 대한 거래량을 가져옵니다.

```
virtual double  CheckOpenLong(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

롱 포지션의 거래량.

참고

이 함수는 미리 정의된 위험 수준이 고정된 롱 포지션에 대한 거래량을 반환합니다. 리스크는 CExpertMoney 기본 클래스의 백분율 매개 변수로 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

단기 포지션에 대한 trade 볼륨을 가져옵니다.

```
virtual double  CheckOpenShort(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

숏 포지션용 거래량.

참고

이 함수는 미리 정의된 위험 수준이 고정된 숏 포지션에 대한 거래량을 반환합니다. 리스크는 CExpertMoney 기본 클래스의 백분율 매개 변수로 정의됩니다.

© 2000-2025, MetaQuotes Ltd.

CMoneyNone

CMoneyNone은 자금 및 리스크 관리의 "부재"를 구현하는 클래스입니다.

Description

CMoneyNone 클래스는 최소 허용 로트로 거래를 구현합니다.

Declaration

```
class CMoneyNone: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneyNone.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertMoney
      CMoneyNone

그룹별 클래스 메서드

Initialization	
virtual ValidationSettings	설정 확인
자금 및 위험 관리 메서드	
virtual CheckOpenLong	롱 포지션에 대한 거래량을 가져옵니다
virtual CheckOpenShort	숏 포지션에 대한 거래량을 가져옵니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

CExpertMoney 클래스에서 상속된 방법

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

성공이면 true, 그렇지 않으면 false.

참고

함수는 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

롱 포지션에 대한 거래량을 가져옵니다.

```
virtual double  CheckOpenLong(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

롱 포지션의 거래량.

참고

함수는 항상 최소 로트 크기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

숏 포지션용 거래량을 가져옵니다.

```
virtual double  CheckOpenShort(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

숏 포지션용 거래량.

참고

함수는 항상 최소 로트 크기를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CMoneySizeOptimized

CMoneySizeOptimized는 이전 거래의 결과에 따라 비용 및 리스크 관리 알고리즘을 구현하는 클래스입니다.

Description

CMoneySizeOptimized는 이전 거래의 결과에 따라 로트 크기로 시장 진입 알고리즘을 구현합니다.

Declaration

```
class CMoneySizeOptimized: public CExpertMoney
```

Title

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

상속 계층

CObject
  CExpertBase
    CExpertMoney
      CMoneySizeOptimized

그룹별 클래스 메서드

Initialization	
DecreaseFactor	매개 변수 값을 설정합니다
virtual ValidationSettings	설정 확인
자금 및 위험 관리 메서드	
virtual CheckOpenLong	롱 포지션에 대한 거래량을 가져옵니다
virtual CheckOpenShort	숏 포지션에 대한 거래량을 가져옵니다

클래스 CObject 에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

CExpertBase 클래스에서 상속된 메서드

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

CExpertMoney 클래스에서 상속된 방법

Percent, CheckReverse, CheckClose

© 2000-2025, MetaQuotes Ltd.

DecreaseFactor

감소 인자의 값을 설정합니다.

```
void  DecreaseFactor(
   double  decrease_factor      // 감소 인자
   )
```

Parameters

decrease_factor

[in] 감소 인자.

참고

DecreaseFactor는 연속 손실 거래의 경우 오프 포지션 볼륨 감소 계수(이전 포지션의 볼륨 대비)를 정의합니다.

© 2000-2025, MetaQuotes Ltd.

ValidationSettings

설정을 확인합니다.

```
virtual bool  ValidationSettings()
```

반환 값

성공이면 true, 그렇지 않으면 false.

참고

감소 요인 값이 음수이면 거짓을 반환하고 그렇지 않으면 참을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenLong

롱 포지션에 대한 거래량을 가져옵니다.

```
virtual double  CheckOpenLong(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

롱 포지션의 거래량.

참고

이 함수는 이전 거래의 결과를 고려하여 장기간 거래량을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

CheckOpenShort

숏 포지션용 거래량을 가져옵니다.

```
virtual double  CheckOpenShort(
   double  price,     // 가격
   double  sl         // 스탑 로스 가격
   )
```

Parameters

price

[in] 추정 시가.

sl

[in] 추정 스탑로스 가격.

반환 값

롱 포지션의 거래량.

참고

이 함수는 이전 거래의 결과를 고려하여 숏 포지션에 대한 거래량을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

제어판 및 대화 상자 만들기 클래스

이 섹션에는 제어판 및 대화 상자 작성을 위한 클래스 작업에 대한 기술 세부 정보와 MQL5 표준 라이브러리의 관련 구성 요소에 대한 설명이 포함되어 있습니다.

이러한 클래스를 사용하면 엑스퍼트 어드바이저 및 지표와 같은 사용자 지정 대화형 MQL5 애플리케이션을 개발할 때 시간을 절약할 수 있습니다.

MQL5 표준 라이브러리(제어판 및 대화 상자 생성을 위한 클래스의 경우)는 터미널 데이터 폴더의 MQL5₩Include₩Controls에 배치됩니다.

다음 문서에서 클래스 작업의 예를 찾아보십시오:

- 모든 복잡성 수준의 그래픽 패널을 생성하는 방법
- 패널 개선: 투명도 추가, 배경색 변경 및 CAppDialog/CWndClient에서 상속
- 지표 또는 엑스퍼트 어드바이저에 제어판 추가
- MQL5에서 자체 그래픽 패널 생성
- 트레이딩을 위해 MQL5에 활성 제어판 생성

이러한 클래스의 작동을 설명하는 엑스퍼트 어드바이저 샘플은 MQL5₩Expert₩Examples₩Controls에서 찾을 수 있습니다.

보조 구조	설명
CRect	Rectangle 영역의 구조
CDateTime	날짜 및 시간으로 작업하는 구조

기본 클래스	설명
CWnd	모든 컨트롤의 기본 클래스
CWndObj	컨트롤 및 대화 상자의 기본 클래스
CWndContainer	복합 컨트롤의 기본 클래스

단순 컨트롤	설명
CLabel	"텍스트 레이블" 그래픽 개체 기반의 컨트롤
CBmpButton	"비트맵 레이블" 그래픽 개체 기반의 컨트롤
CButton	"버튼" 그래픽 개체 기반의 컨트롤
CEdit	"필드 편집" 그래픽 개체를 기반으로 하는 컨트롤
CPanel	"Rectangle 레이블"에 기반한 컨트롤
CPicture	"비트맵 레이블" 기반의 컨트롤

© 2000-2025, MetaQuotes Ltd.

복잡한 제어	설명
CScroll	스크롤 막대 기본 클래스
CScrollV	수직 스크롤 막대
CScrollH	수평 스크롤 막대
CWndClient	스크롤 막대가 있는 클라이언트 영역의 기본 클래스
CListView	목록 보기
CComboBox	콤보 박스
CCheckBox	체크 박스
CCheckGroup	체크 그룹
CRadioButton	라디오 버튼
CRadioGroup	라디오 그룹
CSpinEdit	증가/감소 필드
CDialog	대화 상자
CAppDialog	MQL5 애플리케이션 메인 대화 상자

© 2000-2025, MetaQuotes Ltd.

CRect

CRect는 차트의 직사각형 영역의 클래스입니다.

Description

크렉트는 사각형의 왼쪽 상단 및 오른쪽 하단 모서리의 좌표로 정의되는 영역의 클래스입니다.

Declaration

```
class CRect
```

Title

```
#include <Controls\Rect.mqh>
```

그룹별 클래스 메서드

Properties	
Left	왼쪽 상단 모서리의 X 좌표를 가져오거나 설정합니다
Top	왼쪽 상단 모서리의 Y 좌표를 가져오거나 설정합니다
Right	오른쪽 아래 모서리의 X 좌표를 가져오거나 설정합니다
Bottom	오른쪽 아래 모서리의 Y 좌표를 가져오거나 설정합니다
Width	너비를 가져오거나 설정합니다
Height	높이를 가져오거나 설정합니다
SetBound	영역의 새 좌표를 설정합니다
Move	영역 좌표의 절대 변위를 수행합니다
Shift	영역 좌표의 상대 변위(시프트)를 수행합니다
Contains	점이 영역 내부에 있는지 확인합니다
추가 메서드	
Format	영역 좌표를 문자열로 가져옵니다

© 2000-2025, MetaQuotes Ltd.

Left (Get Method)

왼쪽 상단 모서리의 X 좌표를 가져옵니다.

```
int  Left()
```

Return Value

왼쪽 상단 모서리의 X 좌표.

Left (Set Method)

왼쪽 상단 모서리의 X 좌표를 설정합니다.

```
void  Left(
   const int  x      // x 좌표
   )
```

Parameters

x

[in] 왼쪽 상단 모서리의 새 X 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Top (Get Method)

왼쪽 상단 모서리의 Y 좌표를 가져옵니다.

```
int  Top()
```

Return Value

왼쪽 상단 모서리의 Y 좌표.

Top (Set Method)

왼쪽 상단 모서리의 Y 좌표를 설정합니다.

```
void  Top(
   const int  y      // y 좌표
   )
```

Parameters

y

[in] 왼쪽 상단 모서리의 새 Y 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Right (Get Method)

오른쪽 아래 모서리의 X 좌표를 가져옵니다.

```
int  Right()
```

Return Value

오른쪽 아래 모서리의 X 좌표.

Right (Set Method)

오른쪽 아래 모서리의 X 좌표를 설정합니다.

```
void  Right(
   const int  x      // x 좌표
   )
```

Parameters

x

[in] 오른쪽 아래 모서리의 새 X 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Bottom (Get Method)

오른쪽 아래 모서리의 Y 좌표를 가져옵니다.

```
int  Bottom()
```

Return Value

오른쪽 아래 모서리의 Y 좌표.

Bottom (Set Method)

오른쪽 아래 모서리의 Y 좌표를 설정합니다.

```
void  Bottom(
   const int  y      // y 좌표
   )
```

Parameters

y

[in] 오른쪽 아래 모서리의 새 Y 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Width (Get Method)

영역의 너비를 가져옵니다.

```
int  Width()
```

Return Value

영역의 너비.

Width (Set Method)

영역의 너비를 설정합니다.

```
virtual bool  Width(
   const int  w      // 너비
   )
```

Parameters

w

[in] 새 너비.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Height (Get Method)

영역의 높이를 가져옵니다.

```
int  Height()
```

Return Value

영역의 높이.

Height (Set Method)

영역의 새 높이를 설정합니다.

```
virtual bool  Height(
   const int  h      // 높이
   )
```

Parameters

h

[in] 새 높이.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

SetBound

오른쪽 아래 모서리의 X 좌표를 설정합니다.

```
void  SetBound(
   const & CRect  rect      // CRect 클래스
   )
```

Return Value

None.

SetBound

영역의 새 좌표를 설정합니다.

```
void  SetBound(
   const int  l      // 좌측
   const int  t      // 상단
   const int  r      // 우측
   const int  b      // 하단
   )
```

Parameters

l

[in] 왼쪽 상단 모서리의 X 좌표.

t

[in] 왼쪽 상단 모서리의 Y 좌표.

r

[in] 오른쪽 아래 모서리의 X 좌표.

b

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Move

영역 좌표의 절대 변위를 수행합니다.

```
void  Move(
   const int  x,     // X 좌표
   const int  y      // Y 좌표
   )
```

Parameters

x

[in] 새 X 좌표.

y

[in] 새 Y 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Shift

영역 좌표의 상대 변위(시프트)를 수행합니다.

```
void  Shift(
   const int  dx,     // 델타 X
   const int  dy      // 델타 Y
   )
```

Parameters

dx

[in] 델타 X.

dy

[in] 델타 Y.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Contains

점이 CRect 클래스 영역 내에 있는지 확인합니다.

```
bool  Contains(
   const int  x,     // X 좌표
   const int  y      // Y 좌표
   )
```

Parameters

x

[in] X 좌표.

y

[in] Y 좌표.

Return Value

점이 영역(경계 포함) 내부에 있으면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Format

영역 좌표를 문자열로 가져옵니다.

```
string  Format(
   string &  fmt,    // 포맷
   ) const
```

Parameters

fmt

[in] 형식이 있는 문자열.

Return Value

영역 좌표가 있는 문자열.

© 2000-2025, MetaQuotes Ltd.

CDateTime

CDateTime은 날짜 및 시간으로 작업하기 위한 구조입니다.

Description

CDateTime은 MqlDateTime에서 파생된 구조로 날짜 및 시간을 컨트롤로 사용하는 작업에 사용됩니다.

Declaration

```
struct CDateTime
```

Title

```
#include <Tools\DateTime.mqh>
```

Methods

Properties	
MonthName	월 이름을 가져옵니다
ShortMonthName	월의 짧은 이름을 가져옵니다
DayName	주의 요일의 전체 이름을 가져옵니다
ShortDayName	주의 요일의 단축 이름을 가져옵니다
DaysInMonth	월의 날짜 수를 가져옵니다
메서드 가져오기/설정	
DateTime	날짜와 시간을 가져오기/설정
Date	날짜 설정
Time	시간 설정
Sec	초 설정
Min	분 설정
Hour	시간 설정
Day	월의 날짜 설정
Mon	월 설정
Year	연 설정
추가 메서드	
SecDec	특정 초를 뺍니다
SecInc	특정 초를 더합니다
MinDec	특정 분을 뺍니다

© 2000-2025, MetaQuotes Ltd.

Properties	
MinInc	특정 분을 더합니다
HourDec	특정 시간을 뺍니다
HourInc	특정 시간을 더합니다
DayDec	특정 일 수를 뺍니다
DayInc	특정 일 수를 더합니다
MonDec	특정 월 수를 뺍니다
MonInc	특정 월 수를 더합니다
YearDec	특정 연도 수를 뺍니다
YearInc	특정 연도 수를 더합니다

© 2000-2025, MetaQuotes Ltd.

MonthName

월 이름을 가져옵니다.

```
string  MonthName()  const
```

색인별로 월 이름을 가져옵니다.

```
string  MonthName(
   const int       num          // 월 색인
   )  const
```

Parameters

num

[in] 월 색인 (1-12).

Return Value

전체 월 이름.

© 2000-2025, MetaQuotes Ltd.

ShortMonthName

단축 월 이름을 가져옵니다.

```
string  ShortMonthName()  const
```

색인별 단축 월 이름을 가져옵니다.

```
string  ShortMonthName(
   const int       num          // 월 색인
   )  const
```

Parameters

num

[in] 월 색인 (1-12).

Return Value

단축 월 이름.

© 2000-2025, MetaQuotes Ltd.

DayName

한 주의 전체 요일명을 가져옵니다.

```
string  DayName()  const
```

색인별로 주의 전체 요일 이름을 가져옵니다.

```
string  DayName(
   const int      num         // 요일 색인
   )  const
```

Parameters

num

[in] 요일 색인 (0-6).

Return Value

요일의 전체 이름.

© 2000-2025, MetaQuotes Ltd.

ShortDayName

일주일의 단축 요일명을 가져옵니다.

```
string  ShortDayName()  const
```

인덱스로 단축 요일명을 가져옵니다.

```
string  ShortDayName(
   const int       num          // 요일 색인
   )  const
```

Parameters

num

[in] 요일 색인 (0-6).

Return Value

요일 단축명.

© 2000-2025, MetaQuotes Ltd.

DaysInMonth

월의 일 수를 가져옵니다.

```
int  DaysInMonth()  const
```

Return Value

월의 일 수.

© 2000-2025, MetaQuotes Ltd.

DateTime (Get Method)

날짜와 시간을 가져옵니다.

```
datetime  DateTime()
```

Return Value

datetime 타입의 값.

DateTime (Set Method datetime)

날짜와 시간을 datetime 타입으로 설정합니다.

```
void  DateTime(
   const datetime      value          // 날짜와 시간
   )
```

Parameters

value

[in] datetime 유형의 값.

Return Value

None.

DateTime (Set Method MqlDateTime)

날짜와 시간을 MqlDateTime 타입으로 설정합니다.

```
void  DateTime(
   const MqlDateTime   &value         // 날짜와 시간
   )
```

Parameters

value

[in] MqlDateTime 유형의 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Date (Set Method datetime)

datetime 타입으로 날짜를 설정합니다.

```
void  Date(
   const datetime      value          // 날짜
   )
```

Parameters

value

[in] datetime 유형의 값.

Return Value

None.

Date (Set Method MqlDateTime)

MqlDateTime 타입으로 날짜 설정.

```
void  Date(
   const MqlDateTime   &value         // 날짜
   )
```

Parameters

value

[in] MqlDateTime 유형의 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Time (Set Method datetime)

시간을 datetime 타입으로 설정합니다.

```
void  Time(
   const datetime      value          // 시간
   )
```

Parameters

value

[in] datetime 유형의 값.

Return Value

None.

Time (Set Method MqlDateTime)

시간을 MqlDateTime 타입으로 설정합니다.

```
void  Time(
   const MqlDateTime   &value         // 시간
   )
```

Parameters

value

[in] MqlDateTime 유형의 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Sec

초 설정.

```
void  Sec(
   const int   value        // 초
   )
```

Parameters

value

[in] 초.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Min

분 설정.

```
void  Min(
   const int    value        // 분
   )
```

Parameters

value

[in] 분.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Hour

시간 설정.

```
void  Hour(
   const int    value        // 시간
   )
```

Parameters

value

[in] 시간.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Day

해당 월의 날짜를 설정합니다.

```
void  Day(
   const int   value        // 날짜
   )
```

Parameters

value

[in] 날짜.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Mon

월 설정.

```
void  Mon(
   const int    value        // 월
   )
```

Parameters

value

[in] 월.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Year

연도를 설정합니다.

```
void  Year(
   const int   value        // 연도
   )
```

Parameters

value

[in] 연도.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

SecDec

지정된 시간(초)을 뺍니다.

```
void  SecDec(
   int   delta=1        // 초
   )
```

Parameters

delta

[in] 뺄 시간(초).

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

SecInc

지정된 시간(초)을 추가합니다.

```
void  SecInc(
   int   delta=1        // 초
   )
```

Parameters

delta

[in] 추가 시간(초).

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

MinDec

지정된 시간(분)을 뺍니다.

```
void  MinDec(
   int   delta=1        // 분
   )
```

Parameters

delta

[in] 뺄 시간(분).

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

MinInc

지정된 시간(분)을 추가합니다.

```
void  MinInc(
   int   delta=1        // 분
   )
```

Parameters

delta

[in] 추가할 시간(분).

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

HourDec

지정된 시간을 뺍니다.

```
void  HourDec(
   int   delta=1        // 시간
   )
```

Parameters

delta

[in] 뺄 시간.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

HourInc

지정된 시간을 추가합니다.

```
void  HourInc(
   int   delta=1        // 시간
   )
```

Parameters

delta

[in] 추가할 시간.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

DayDec

특정 일 수를 뺍니다.

```
void  DayDec(
   int   delta=1        // 일
   )
```

Parameters

delta

[in] 뺄 날짜 수.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

DayInc

특정 일 수를 추가합니다.

```
void  DayInc(
   int   delta=1        // 일
   )
```

Parameters

delta

[in] 추가할 일 수.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

MonDec

지정한 개월 수를 뺍니다.

```
void  MonDec(
   int   delta=1        // 월
   )
```

Parameters

delta

[in] 뺄 월 수.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

MonInc

지정한 개월 수를 추가합니다.

```
void  MonInc(
   int   delta=1        // 월
   )
```

Parameters

delta

[in] 추가할 월수.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

YearDec

지정된 연 수를 뺍니다.

```
void  YearDec(
   int   delta=1        // 연 수
   )
```

Parameters

delta

[in] 뺄 연 수.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

YearInc

지정된 연 수를 추가합니다.

```
void  YearInc(
   int   delta=1        // 연 수
   )
```

Parameters

delta

[in] 추가할 연 수.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

CWnd

CWnd는 모든 표준 라이브러리 컨트롤의 기본 클래스입니다.

Description

CWnd 클래스는 기본 제어 클래스의 구현입니다.

Declaration

```
class CWnd : public CObject
```

Title

```
#include <Controls\Wnd.mqh>
```

상속 계층

CObject
  CWnd

직계 종속

CDragWnd, CWndContainer, CWndObj

그룹별 클래스 메서드

생성 및 제거	
Create	컨트롤 생성
Destroy	컨트롤 제거
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
OnMouseEvent	CHARTEVENT_MOUSE_MOVE 이벤트의 이벤트 핸들러
Name	
Name	컨트롤 명을 가져옵니다
컨테이너에 대한 액세스	
ControlsTotal	컨테이너의 컨트롤 수를 가져옵니다
Control	색인별로 컨트롤을 가져옵니다
ControlFind	ID별로 컨트롤을 가져옵니다
Geometry	
Rect	컨트롤 좌표를 가져옵니다
Left	왼쪽 상단 모서리의 X 좌표를 가져오거나 설정합니다

© 2000-2025, MetaQuotes Ltd.

생성 및 제거	
Top	왼쪽 상단 모서리의 Y 좌표를 가져오거나 설정합니다
Right	오른쪽 아래 모서리의 X 좌표를 가져오거나 설정합니다
Bottom	오른쪽 아래 모서리의 Y 좌표를 가져오거나 설정합니다
Width	컨트롤 너비를 가져오거나 설정합니다
Height	컨트롤 높이를 가져오거나 설정합니다
Move	절대 변위를 제어합니다
Shift	상대 변위를 제어합니다
Resize	컨트롤 규모를 변경합니다
Contains	점/컨트롤이 제어 영역 내부에 있는지 점검합니다
Align	
Alignment	컨트롤의 선형 특성을 설정합니다
Align	지정된 차트 영역에서 컨트롤 정렬을 수행합니다
Identification	
Id	컨트롤 ID를 가져오거나 설정합니다
State	
IsEnabled	사용자의 작업에 응답할 수 있는 제어 능력을 점검합니다
Enable	사용자의 작업에 응답할 수 있는 제어 기능을 사용합니다
Disable	사용자의 작업에 응답하는 제어 기능을 사용 불가능으로 설정합니다
IsVisible	가시성 플래그를 확인합니다
Visible	가시성 플래그를 확인합니다
Show	컨트롤을 표시합니다
Hide	컨트롤을 숨깁니다
IsActive	컨트롤 활동을 점검합니다
Activate	컨트롤을 활성화합니다
Deactivate	컨트롤을 비활성화합니다
상태 플래그	
StateFlags	상태 플래그 컨트롤을 가져오거나 설정합니다
StateFlagsSet	컨트롤 상태 플래그를 설정합니다
StateFlagsReset	컨트롤 상태 플래그를 재설정합니다
속성 플래그	

© 2000-2025, MetaQuotes Ltd.

생성 및 제거	
PropFlags	컨트롤 속성 플래그를 가져오거나 재설정합니다
PropFlagsSet	컨트롤 속성 플래그를 설정합니다
PropFlagsReset	컨트롤 속성 플래그를 재설정합니다
마우스 작업	
MouseX	마우스 X 좌표를 가져오거나 저장합니다
MouseY	마우스 Y 좌표를 가져오거나 저장합니다
MouseFlags	마우스 버튼 상태를 가져오거나 저장합니다
MouseFocusKill	마우스 포커스를 죽입니다
내부 이벤트 핸들러	
OnCreate	"Create" 이벤트 핸들러
OnDestroy	"Destroy" 이벤트 핸들러
OnMove	"Move" 이벤트 핸들러
OnResize	"Resize" 이벤트 핸들러
OnEnable	"Enable" 이벤트 핸들러
OnDisable	"Disable" 이벤트 핸들러
OnShow	"Show" 이벤트 핸들러
OnHide	"Hide" 이벤트 핸들러
OnActivate	"Activate" 이벤트 핸들러
OnDeactivate	"Deactivate" 이벤트 핸들러
OnClick	"Click" 이벤트 핸들러
OnChange	"Change" 이벤트 핸들러
마우스 이벤트 핸들러	
OnMouseDown	"MouseDown" 이벤트 핸들러
OnMouseUp	"MouseUp" 이벤트 핸들러
이벤트 핸들러 드래그	
OnDragStart	"DragStart" 이벤트 핸들러
OnDragProcess	"DragProcess" 이벤트 핸들러
OnDragEnd	"DragEnd" 이벤트 핸들러
드래그 객체	
DragObjectCreate	드래그 객체를 생성합니다

© 2000-2025, MetaQuotes Ltd.

생성 및 제거	
DragObjectDestroy	드래그 객체를 삭제합니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

© 2000-2025, MetaQuotes Ltd.

Create

컨트롤을 생성합니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // x1 좌표
   const int      y1,         // y1 좌표
   const int      x2,         // x2 좌표
   const int      y2          // y2 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트 하위 창.

x1

[in] 왼쪽 상단 모서리의 X 좌표.

y1

[in] 왼쪽 상단 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

Note

기본 클래스 메서드는 작성 매개 변수만 저장하고 항상 true를 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Destroy

컨트롤을 제거합니다.

```
virtual bool  Destroy()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 이벤트 파라미터
   const double&  dparam,      // 이벤트 파라미터
   const string&  sparam       // 이벤트 파라미터
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조에 의해 전달된 long 유형의 이벤트 매개 변수.

dparam

[in] 참조에 의해 전달된 double 유형의 이벤트 매개 변수.

sparam

[in] 참조에 의해 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되었다면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

OnMouseEvent

마우스 이벤트 핸들러(CHARTEVENT_MOUSE_MOVE 차트 이벤트).

```
virtual bool  OnMouseEvent(
   const int  x,          // x 좌표
   const int  y,          // y 좌표
   const int  flags       // 플래그
   )
```

Parameters

x

[in] 차트의 왼쪽 상단 모서리를 기준으로 한 마우스 커서의 X 좌표.

y

[in] 차트의 왼쪽 상단 모서리를 기준으로 한 마우스 커서의 Y 좌표.

flags

[in] 마우스 버튼 상태 플래그.

Return Value

이벤트가 처리되었다면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Name

컨트롤 명을 가져옵니다.

```
string  Name()  const
```

Return Value

컨트롤명.

© 2000-2025, MetaQuotes Ltd.

ControlsTotal

컨테이너의 컨트롤 수를 가져옵니다.

```
int  ControlsTotal()  const
```

Return Value

컨테이너의 컨트롤 수.

Note

기본 클래스 메서드는 컨테이너가 없으며 상속자에게 컨테이너에 대한 액세스를 제공하고 항상 0을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Control

색인별로 컨테이너에서 컨트롤을 가져옵니다.

```
CWnd*  Control(
   const int  ind      // 인덱스
   )  const
```

Parameters

ind

[in] 컨트롤 인덱스.

Return Value

컨트롤에 대한 포인터.

Note

기본 클래스 메서드는 컨테이너가 없으며 상속자에게 컨테이너에 대한 액세스를 제공하고 항상 NULL을 반환합니다.

© 2000-2025, MetaQuotes Ltd.

ControlFind

지정된 ID별로 컨테이너에서 컨트롤을 가져옵니다.

```
virtual CWnd*  ControlFind(
   const long  id      // ID
   )
```

Parameters

id

[in] 찾을 컨트롤의 식별자.

Return Value

컨테이너에서 컨트롤에 대한 포인터.

Note

기본 클래스 메서드는 컨테이너가 없으며 상속자에게 컨테이너에 대한 액세스를 제공합니다. 지정된 ID가 컨테이너의 ID와 일치하면 포인터를 자신(this)에게 반환합니다.

© 2000-2025, MetaQuotes Ltd.

Rect

CRect クラス 객체에 대한 포인터를 가져옵니다.

```
const CRect*  Rect()  const
```

Return Value

CRect 클래스 객체에 대한 포인터.

© 2000-2025, MetaQuotes Ltd.

Left (Get Method)

컨트롤의 왼쪽 상단 모서리에 대한 X 좌표를 가져옵니다.

```
int  Left()
```

Return Value

컨트롤의 왼쪽 상단 모서리의 X 좌표.

Left (Set Method)

컨트롤의 왼쪽 상단 모서리에 대한 X 좌표를 설정합니다.

```
void  Left(
   const int  x      // 좌표
   )
```

Parameters

x

[in] 왼쪽 상단 모서리의 새 X 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Top (Get Method)

컨트롤의 왼쪽 상단 모서리의 Y 좌표를 가져옵니다.

```
int  Top()
```

Return Value

컨트롤의 왼쪽 상단 모서리의 Y 좌표.

Top (Set Method)

컨트롤의 왼쪽 상단 모서리에 대한 Y 좌표를 설정합니다.

```
void  Top(
   const int  y      // y 좌표
   )
```

Parameters

y

[in] 왼쪽 상단 모서리의 새 Y 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Right (Get Method)

컨트롤의 오른쪽 하단 모서리의 X 좌표를 가져옵니다.

```
int  Right()
```

Return Value

오른쪽 아래 모서리의 X 좌표.

Right (Set Method)

컨트롤의 오른쪽 하단 모서리의 X 좌표를 설정합니다.

```
void  Right(
   const int  x      // x 좌표
   )
```

Parameters

x

[in] 오른쪽 아래 모서리의 새 X 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Bottom (Get Method)

컨트롤의 오른쪽 하단 모서리의 Y 좌표를 가져옵니다.

```
int  Bottom()
```

Return Value

컨트롤의 오른쪽 하단 모서리에 있는 Y 좌표.

Bottom (Set Method)

컨트롤의 오른쪽 하단 모서리의 Y 좌표를 설정합니다.

```
void  Bottom(
   const int  y      // y 좌표
   )
```

Parameters

y

[in] 오른쪽 아래 모서리의 새 Y 좌표.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Width (Get Method)

컨트롤 너비를 가져옵니다.

```
int  Width()
```

Return Value

컨트롤의 너비.

Width (Set Method)

컨트롤의 너비를 설정합니다.

```
virtual bool  Width(
   const int  w      // 너비
   )
```

Parameters

w

[in] 새 너비.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Height (Get Method)

컨트롤 높이를 가져옵니다.

```
int  Height()
```

Return Value

컨트롤 높이.

Height (Set Method)

컨트롤의 새 높이를 설정합니다.

```
virtual bool  Height(
   const int  h      // 높이
   )
```

Parameters

h

[in] 새 높이.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Move

컨트롤의 절대 변위를 수행합니다.

```
void  Move(
   const int  x,     // x 좌표
   const int  y      // y 좌표
   )
```

Parameters

x

[in] 왼쪽 상단 점의 새 X 좌표.

y

[in] 왼쪽 상단 점의 새 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Shift

컨트롤의 상대적 시프트를 수행합니다.

```
void  Shift(
   const int  dx,     // 델타 X
   const int  dy      // 델타 Y
   )
```

Parameters

dx

[in] 델타 X.

dy

[in] 델타 Y.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Resize

컨트롤의 크기를 설정합니다.

```
virtual bool  Resize(
   const int  w,
   const int  h
   )
```

Parameters

w

[in] 새 너비.

h

[in] 새 높이.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Contains

점이 차트의 관리 영역 안에 있는지 확인합니다.

```
bool  Contains(
   const int  x,     // X 좌표
   const int  y      // Y 좌표
   )
```

Parameters

x

[in] X 좌표.

y

[in] Y 좌표.

Return Value

점이 영역(경계 포함) 내부에 있으면 true, 그렇지 않으면 false.

Contains

지정된 컨트롤이 차트의 제어 영역 안에 있는지 확인합니다.

```
bool  Contains(
   const CWnd*  control      // 포인터
   )  const
```

Parameters

control

[in] 객체 포인터.

Return Value

지정된 컨트롤이 영역(경계 포함) 안에 있으면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Alignment

컨트롤의 선형 매개변수를 설정합니다.

```
void  Alignment(
   const int  flags,       // 플래그
   const int  left,        // 오프셋
   const int  top,         // 오프셋
   const int  right,       // 오프셋
   const int  bottom       // 오프셋
   )
```

Parameters

flags

[in] 정렬 플래그.

left

[in] 왼쪽 테두리로부터의 고정 오프셋.

top

[in] 상단 테두리에서 고정 오프셋.

right

[in] 오른쪽 테두리로부터의 고정 오프셋.

bottom

[in] 하단 테두리에서 고정 오프셋.

Return Value

None.

Note

플래그 정렬:

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                  // 정렬 없음
   WND_ALIGN_LEFT=1,                                  // 왼쪽 정렬
   WND_ALIGN_TOP=2,                                   // 상단 정렬
   WND_ALIGN_RIGHT=4,                                 // 우측 정렬
   WND_ALIGN_BOTTOM=8,                                // 하단 정렬
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,  // 정렬 너비
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,   // 정렬 높이
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // 높이 및 너비 정렬
   }
```

© 2000-2025, MetaQuotes Ltd.

Align

지정된 차트 영역에서 제어 정렬을 수행합니다.

```
virtual bool  Align(
   const CRect*  rect      // 포인터
   )
```

Parameters

rect

[in] 차트 영역 좌표가 있는 객체에 대한 포인터.

Return Value

성공시 true, 아니면 false.

Note

선형 매개변수를 지정해야 합니다(기본적으로 선형 없음).

© 2000-2025, MetaQuotes Ltd.

Id (Get Method)

컨트롤 ID를 가져옵니다.

```
long  Id()  const
```

Return Value

컨트롤 식별자.

Id (Set Method)

컨트롤 ID 값을 설정합니다.

```
virtual long  Id(
   const long  id      // 식별자
   )
```

Parameters

id

[in] 컨트롤 식별자의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

IsEnabled

사용자의 작업에 응답할 수 있는 컨트롤 능력을 점검합니다.

```
bool  IsEnabled()  const
```

Return Value

컨트롤이 활성화되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Enable

사용자의 작업에 응답할 수 있는 제어 기능을 사용합니다.

```
virtual bool  Enable()
```

Return Value

성공시 true, 아니면 false.

Note

컨트롤이 활성화된 경우 외부 이벤트를 처리할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

Disable

사용자의 작업에 응답하는 제어 기능을 사용 불가능으로 설정합니다.

```
virtual bool  Disable()
```

Return Value

성공시 true, 아니면 false.

Note

비활성화된 컨트롤은 외부 이벤트를 처리할 수 없습니다.

© 2000-2025, MetaQuotes Ltd.

IsVisible

컨트롤이 표시되는지 여부를 확인합니다.

```
bool  IsVisible()  const
```

Return Value

컨트롤이 차트에 표시되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Visible

가시성 플래그를 설정합니다.

```
virtual bool  Visible(
   const bool  flag      // 플래그
   )
```

Parameters

flag

[in] 새 플래그.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Show

컨트롤 표시.

```
virtual bool  Show()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Hide

컨트롤을 숨깁니다.

```
virtual bool  Hide()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

IsActive

컨트롤이 활성 상태인지 여부를 나타내는 값을 가져옵니다.

```
bool  IsActive()  const
```

Return Value

컨트롤이 활성화 상태면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

Activate

컨트롤을 활성화합니다

```
virtual bool  Activate()
```

Return Value

성공시 true, 아니면 false.

Note

마우스 커서 그것 위에 떠 있는 제어 활발히 진행됩니다.

© 2000-2025, MetaQuotes Ltd.

Deactivate

컨트롤을 비활성화합니다.

```
virtual bool  Deactivate()
```

Return Value

성공시 true, 아니면 false.

Note

마우스 커서가 컨트롤에서 벗어나면 컨트롤이 비활성화됩니다.

© 2000-2025, MetaQuotes Ltd.

StateFlags (Get Method)

컨트롤 상태 플래그를 가져옵니다.

```
int  StateFlags()
```

Return Value

컨트롤 상태 플래그.

StateFlags (Set Method)

컨트롤 상태 플래그를 설정합니다.

```
virtual void  StateFlags(
   const int  flags      // 플래그
   )
```

Parameters

flags

[in] 새 컨트롤 상태 플래그.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

StateFlagsSet

컨트롤 상태 플래그를 설정합니다.

```
virtual void  StateFlagsSet(
   const int  flags      // 플래그
   )
```

Parameters

flags

[in] 설정할 플래그.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

StateFlagsReset

컨트롤 상태 플래그를 재설정합니다.

```
virtual void  StateFlagsReset(
   const int  flags      // 플래그
   )
```

Parameters

flags

[in] 재설정할 플래그.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

PropFlags (Get Method)

컨트롤 속성 플래그를 가져옵니다.

```
void  PropFlags(
   const int  flags      // 플래그
   )
```

Return Value

컨트롤 속성 플래그.

PropFlags (Set Method)

컨트롤 속성 플래그를 설정합니다.

```
virtual void  PropFlags(
   const int  flags      // 플래그
   )
```

Parameters

flags

[in] 새 플래그.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

PropFlagsSet

컨트롤 속성 플래그를 설정합니다.

```
virtual void  PropFlagsSet(
   const int  flags      // 플래그
   )
```

Parameters

flags

[in] 설정할 플래그.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

PropFlagsReset

컨트롤 속성 플래그를 재설정합니다.

```
virtual void  PropFlagsReset(
   const int  flags      // 플래그
   )
```

Parameters

flags

[in] 재설정할 플래그.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

MouseX (Set Method)

마우스 X 좌표를 저장합니다.

```
void  MouseX(
   const int  value      // 좌표
   )
```

Parameters

value

[in] 마우스의 X 좌표.

Return Value

None.

MouseX (Get Method)

마우스의 저장된 X 좌표를 가져옵니다.

```
int  MouseX()
```

Return Value

마우스의 X 좌표.

© 2000-2025, MetaQuotes Ltd.

MouseY (Set Method)

마우스 Y 좌표를 저장합니다.

```
void  MouseY(
   const int  value      // 좌표
   )
```

Parameters

value

[in] 마우스의 Y 좌표.

Return Value

None.

MouseY (Get Method)

마우스의 저장된 Y 좌표를 가져옵니다.

```
int  MouseY()
```

Return Value

마우스의 Y 좌표.

© 2000-2025, MetaQuotes Ltd.

MouseFlags (Set Method)

마우스 버튼의 상태를 저장합니다.

```
virtual void  MouseFlags(
   const int  value      // 상태
   )
```

Parameters

value

[in] 마우스 버튼 상태.

Return Value

None.

MouseFlags (Get Method)

마우스 버튼의 저장된 상태를 가져옵니다.

```
int  MouseFlags()
```

Return Value

마우스 버튼의 상태.

© 2000-2025, MetaQuotes Ltd.

MouseFocusKill

마우스 버튼의 저장된 상태를 지우고 컨트롤을 비활성화합니다.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // id
   )
```

Parameters

id=CONTROLS_INVALID_ID

[in] 마우스 포커스를 수신한 컨트롤의 식별자.

Return Value

컨트롤 비활성화 결과.

© 2000-2025, MetaQuotes Ltd.

OnCreate

내부 "Create" 이벤트의 가상 핸들러.

```
virtual bool  OnCreate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnDestroy

내부 "Destroy" 이벤트의 가상 핸들러.

```
virtual bool  OnDestroy()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnMove

내부 "Move" 이벤트의 가상 핸들러.

```
virtual bool  OnMove()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnResize

내부 "Resize" 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnEnable

내부 "Enable" 이벤트의 가상 핸들러(사용 가능한 경우 사용자 상호 작용에 응답할 수 있음).

```
virtual bool  OnEnable()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnDisable

내부 "사용 안 함" 이벤트의 가상 핸들러(사용 안 함으로 설정된 경우 사용자 상호 작용에 응답할 수 없음).

```
virtual bool  OnDisable()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnShow

내부 "Show" 이벤트의 가상 처리기.

```
virtual bool  OnShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnHide

내부 "Hide" 이벤트의 가상 핸들러.

```
virtual bool  OnHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnActivate

내부 "Activate" 이벤트의 가상 핸들러.

```
virtual bool  OnActivate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

내부 "활성화 해제" 이벤트의 가상 핸들러.

```
virtual bool  OnDeactivate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnClick

내부 "Click" 이벤트의 가상 핸들러(마우스 클릭).

```
virtual bool  OnClick()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChange

내부 "Change" 이벤트의 가상 핸들러.

```
virtual bool  OnChange()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

"마우스 다운"(마우스 클릭) 이벤트의 가상 핸들러.

```
virtual bool  OnMouseDown()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

컨트롤에서 마우스 왼쪽 버튼을 누르면 "MouseDown" 이벤트가 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

"MouseUp"(마우스 단추 해제) 이벤트의 가상 핸들러.

```
virtual bool  OnMouseUp()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

컨트롤 "MouseUp"(마우스 왼쪽 버튼 해제) 이벤트의 가상 핸들러.

© 2000-2025, MetaQuotes Ltd.

OnDragStart

"DragStart"(끌기 시작) 이벤트의 가상 핸들러.

```
virtual bool  OnDragStart()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"DragStart" 이벤트는 제어 끌기 작업이 시작될 때 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnDragProcess

"DragProcess"(끌기) 이벤트의 가상 핸들러.

```
virtual bool  OnDragProcess(
   const int  x,      // x 좌표
   const int  y       // y 좌표
   )
```

Parameters

x

[in] 마우스 커서의 현재 X 좌표.

y

[in] 마우스 커서의 현재 Y 좌표.

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

컨트롤을 끌면 "DragProcess" 이벤트가 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnDragEnd

"DragEnd" 이벤트의 가상 핸들러.

```
virtual bool  OnDragEnd()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

제어 드래그 프로세스가 완료되면 "DragEnd" 이벤트가 발생합니다.

© 2000-2025, MetaQuotes Ltd.

DragObjectCreate

드래그 객체를 생성합니다.

```
virtual bool  DragObjectCreate()
```

Return Value

성공시 true, 아니면 false.

Note

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

DragObjectDestroy

드래그 객체를 제거합니다.

```
virtual bool  DragObjectDestroy()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CWndObj

CWndObj는 표준 라이브러리의 단순 컨트롤(차트 객체 기준)

Description

CWndObj 클래스는 단순 컨트롤의 기본 메소드를 구현합니다.

Declaration

```
class CWndObj : public CWnd
```

Title

```
#include <Controls\WndObj.mqh>
```

상속 계층

CObject
  CWnd
    CWndObj

직계 종속

CBmpButton, CButton, CEdit, CLabel, CPanel, CPicture

그룹별 클래스 메서드

차트 이벤트 처리	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Properties	
Text	차트 객체의 OBJPROP_TEXT 속성을 가져오거나 설정합니다
Color	차트 객체의 OBJPROP_COLOR 속성을 가져오거나 설정합니다
ColorBackground	차트 객체의 OBJPROP_BGCOLOR 속성을 가져오거나 설정합니다
ColorBorder	차트 객체의 OBJPROP_BORDER_COLOR 속성을 가져오거나 설정합니다
Font	차트 객체의 OBJPROP_FONT 속성을 가져오거나 설정합니다
FontSize	차트 객체의 OBJPROP_FONTSIZE 속성을 가져오거나 설정합니다
ZOrder	차트 객체의 OBJPROP_ZORDER 속성을 가져오거나 설정합니다
차트 객체 이벤트 핸들러	
OnObjectCreate	CHARTEVENT_OBJECT_CREATE 이벤트 핸들러
OnObjectChange	CHARTEVENT_OBJECT_CHANGE 이벤트 핸들러
OnObjectDelete	CHARTEVENT_OBJECT_DELETE이벤트 핸들러

© 2000-2025, MetaQuotes Ltd.

차트 이벤트 처리	
OnObjectDrag	CHARTEVENT_OBJECT_DRAG 이벤트 핸들러
속성 변경 이벤트 핸들러	
OnSetText	"SetText" 이벤트 핸들러
OnSetColor	"SetColor" 이벤트 핸들러
OnSetColorBackground	"SetColorBackground" 이벤트 핸들러
OnSetFont	"SetFont" 이벤트 핸들러
OnSetFontSize	"SetFontSize" 이벤트 핸들러
OnSetZOrder	"SetZOrder" 이벤트 핸들러
내부 이벤트 핸들러	
OnDestroy	"Destroy" 내부 이벤트 ㅎ핸들러
OnChange	내부 이벤트 핸들러 "Change"

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CWnd에서 상속된 메서드

Create, Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조에 의해 전달된 long 유형의 이벤트 매개 변수.

dparam

[in] 참조에 의해 전달된 double 유형의 이벤트 매개 변수.

sparam

[in] 참조에 의해 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Text (Get method)

차트 객체의 OBJPROP_TEXT(텍스트) 속성을 가져옵니다.

```
string  Text()
```

Return Value

OBJPROP_TEXT 속성의 값.

Text (Set method)

차트 객체의 OBJPROP_TEXT(텍스트) 속성을 설정합니다.

```
bool  Text(
   const string  value      // 값
   )
```

Parameters

value

[in] OBJPROP_TEXT 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Color (Get method)

차트 객체의 OBJPROP_COLOR X(색상) 특성을 가져옵니다.

```
color  Color()
```

Return Value

OBJPROP_COLOR 속성의 값.

Color (Set method)

차트 객체의 OBJPROP_COLOR(색상) 속성을 설정합니다.

```
bool  Color(
   const color  value      // 값
   )
```

Parameters

value

[in] OBJPROP_COLOR 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ColorBackground (Get method)

차트 객체의 OBJPROP_BGCOLOR(배경색)을 가져옵니다.

```
color  ColorBackground()
```

Return Value

OBJPROP_BGCOLOR 속성의 값.

ColorBackground (Set method)

차트 객체의 OBJPROP_BGCOLOR (배경색) 속성을 설정합니다.

```
bool  ColorBackground(
   const color  value      // 값
   )
```

Parameters

value

[in] OBJPROP_BGCOLOR 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ColorBorder (Get method)

차트 객체의 OBJPROP_BORDER_COLOR(경계 색상) 특성을 가져옵니다.

```
color  ColorBorder()
```

Return Value

OBJPROP_BORDER_COLOR 속성의 값.

ColorBorder (Set method)

차트 객체의 OBJPROP_BORDER_COLOR(테두리 색상) 특성을 설정합니다.

```
bool  ColorBorder(
   const color  value      // 값
   )
```

Parameters

value

[in] New value of the OBJPROP_BORDER_COLOR property.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Font (Get method)

차트 객체의 OBJPROP_FONT(font) 속성을 가져옵니다.

```
string  Font()
```

Return Value

OBJPROP_FONT 특성의 값.

Font (Set method)

차트 객체의 OBJPROP_FONT(font) 속성을 설정합니다.

```
bool  Font(
   const string  value      // 값
   )
```

Parameters

value

[in] OBJPROP_FONT 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

FontSize (Get method)

차트 객체의 OBJPROP_FONTSIZE(글꼴 크기) 속성을 가져옵니다.

```
int  FontSize()
```

Return Value

OBJPROP_FONTSIZE 속성의 값.

FontSize (Set method)

차트 객체의 OBJPROP_FONTSIZE(글꼴 크기) 속성을 설정합니다.

```
bool  FontSize(
   const int  value      // 값
   )
```

Parameters

value

[in] OBJPROP_FONTSIZE 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ZOrder (Get method)

차트 객체의 OBJPROP_ZORDER 속성을 가져옵니다.

```
long  ZOrder()
```

Return Value

OBJPROP_ZORDER 속성의 값.

ZOrder (Set method)

차트 객체의 OBJPROP_ZORDER 속성을 설정합니다.

```
bool  ZOrder(
   const long  value      // 값
   )
```

Parameters

value

[in] OBJPROP_ZORDER 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnObjectCreate

차트 객체 CHARTEVENT_OBJECT_CREATE 이벤트의 가상 핸들러.

```
virtual bool  OnObjectCreate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnObjectChange

차트 객체 CHARTEVENT_OBJECT_CHANGE 이벤트의 가상 핸들러.

```
virtual bool  OnObjectChange()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnObjectDelete

차트 객체 CHARTEVENT_OBJECT_DELETE 이벤트의 가상 핸들러.

```
virtual bool  OnObjectDelete()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnObjectDrag

차트 객체 CHARTEVENT_OBJECT_DRAG 이벤트의 가상 핸들러.

```
virtual bool  OnObjectDrag()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetText

컨트롤 "SetText" (OBJPROP_TEXT 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetText()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

컨트롤 "SetColor" (OBJPROP_COLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColor()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

컨트롤 "SetColorBackground" (OBJPROP_BGCOLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColorBackground()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

"SetFont" (OBJPROP_FONT 속성 변화) 이벤트의 가상 핸들러.

```
virtual bool  OnSetFont()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

"SetFontSize" (OBJPROP_FONTSIZE 속성 변화) 이벤트의 가상 핸들러.

```
virtual bool  OnSetFontSize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

"SetZOrder" (OBJPROP_ZORDER 속성 변화) 이벤트의 가상 핸들러.

```
virtual bool  OnSetZOrder()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnDestroy

제어 "Destroy" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnDestroy()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChange

컨트롤 "Change" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChange()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CWndContainer

CWndContainer는 표준 라이브러리의 복합 컨트롤(종속 컨트롤 포함)을 위한 기본 클래스입니다.

Description

CWndContainer 클래스는 복합 컨트롤의 기본 메서드를 구현합니다.

Declaration

```
class CWndContainer : public CWnd
```

Title

```
#include <Controls\WndContainer.mqh>
```

상속 계층

CObject
  CWnd
    CWndContainer

직계 종속

CCheckBox, CComboBox, CDateDropList, CDatePicker, CDialog, CRadioButton, CScroll, CSpinEdit, CWndClient

그룹별 클래스 메서드

Destroy	
Destroy	모든 컨테이너 컨트롤을 폐기합니다
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
OnMouseEvent	The CHARTEVENT_MOUSE_MOVE event handler
컨테이너에 대한 액세스	
ControlsTotal	컨테이너의 컨트롤 수를 가져옵니다
Control	색인별 컨트롤을 가져옵니다
ControlFind	ID별 컨트롤을 가져옵니다
Add/Delete	
Add	그룹(컨테이너)에 컨트롤을 추가합니다
Delete	그룹(컨테이너)에서 컨트롤을 삭제합니다
Geometry	
Move	요소 그룹의 절대 변위를 수행합니다

© 2000-2025, MetaQuotes Ltd.

Destroy	
Shift	요소 그룹의 상대 변위를 수행합니다
Identification	
Id	컨테이너의 모든 컨트롤에 대한 ID를 설정합니다
State	
Enable	컨테이너의 모든 컨트롤을 활성화합니다
Disable	컨테이너의 모든 컨트롤을 비활성화합니다
Show	용기의 모든 컨트롤을 표시합니다
Hide	용기의 모든 컨트롤을 숨깁니다
마우스 작업	
MouseFocusKill	마우스 포커스를 죽입니다
파일 작업	
Save	컨테이너 정보를 파일에 저장합니다
Load	파일에서 컨테이너 정보를 로드합니다
내부 이벤트 핸들러	
OnResize	"Resize" 이벤트 핸들러
OnActivate	"Activate" 이벤트 핸들러
OnDeactivate	"Deactivate" 이벤트 핸들러

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Create, Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

© 2000-2025, MetaQuotes Ltd.

Destroy

모든 컨테이너 컨트롤을 폐기합니다.

```
virtual bool  Destroy()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조에 의해 전달된 long 유형의 이벤트 매개 변수.

dparam

[in] 참조에 의해 전달된 double 유형의 이벤트 매개 변수.

sparam

[in] 참조에 의해 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMouseEvent

마우스 이벤트 핸들러.

```
virtual bool  OnMouseEvent(
   const int  x,          // x 좌표
   const int  y,          // y 좌표
   const int  flags       // 플래그
   )
```

Parameters

x

[in] 차트의 왼쪽 상단 모서리를 기준으로 한 마우스 커서의 X 좌표.

y

[in] 차트의 왼쪽 상단 모서리를 기준으로 한 마우스 커서의 Y 좌표.

flags

[in] 마우스 버튼 상태 플래그.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ControlsTotal

컨테이너의 컨트롤 수를 가져옵니다.

```
int  ControlsTotal()  const
```

Return Value

컨테이너 안의 컨트롤 수.

© 2000-2025, MetaQuotes Ltd.

Control

인덱스로 컨테이너에서 컨트롤을 가져옵니다.

```
CWnd*  Control(
   const int  ind      // 인덱스
   )  const
```

Parameters

ind

[in] 필요한 컨트롤 색인.

Return Value

컨트롤에 대한 포인터, 컨트롤을 찾을 수 없는 경우 NULL.

© 2000-2025, MetaQuotes Ltd.

ControlFind

식별자로 컨테이너에서 제어를 가져옵니다.

```
virtual CWnd*  ControlFind(
   const long  id      // id
   )
```

Parameters

id

[in] 컨트롤 ID.

Return Value

컨트롤에 대한 포인터, 컨트롤을 찾을 수 없는 경우 NULL.

© 2000-2025, MetaQuotes Ltd.

Add

그룹(컨테이너)에 컨트롤을 추가합니다.

```
bool  Add(
   CWnd&  control      // 참조
   )
```

Parameters

control

[in] 추가할 컨트롤, 참조에 의해 전달됨.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Delete

그룹(컨테이너)에서 컨트롤을 삭제합니다.

```
bool  Delete(
   CWnd&  control      // 참조
   )
```

Parameters

control

[in] 삭제할 컨트롤, 참조에 의해 전달됨.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Move

컨테이너의 모든 컨트롤의 절대 변위를 수행합니다.

```
virtual bool  Move(
   const int  x,     // X 좌표
   const int  y      // Y 좌표
   )
```

Parameters

x

[in] 왼쪽 상단 모서리의 새 X 좌표.

y

[in] 왼쪽 상단 모서리의 새 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Shift

용기의 모든 컨트롤에 대해 좌표의 상대적 이동을 수행합니다.

```
virtual bool  Shift(
   const int  dx,     // 델타 X
   const int  dy      // 델타 Y
   )
```

Parameters

dx

[in] 델타 X.

dy

[in] 델타 Y.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Id

컨테이너의 모든 컨트롤에 대한 ID를 설정합니다.

```
virtual long  Id(
   const long  id      // 식별자
   )
```

Parameters

id

[in] 기본 그룹 식별자.

Return Value

컨테이너 컨트롤에서 사용되는 식별자 수.

© 2000-2025, MetaQuotes Ltd.

Enable

컨테이너의 모든 컨트롤을 활성화합니다.

```
virtual bool  Enable()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Disable

컨테이너의 모든 컨트롤을 비활성화합니다.

```
virtual bool  Disable()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Show

컨테이너의 모든 컨트롤을 표시합니다.

```
virtual bool  Show()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Hide

컨테이너의 모든 컨트롤을 숨깁니다.

```
virtual bool  Hide()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

MouseFocusKill

마우스 버튼의 저장된 상태를 지우고 용기의 모든 컨트롤을 비활성화합니다.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // id
   )
```

Parameters

id=CONTROLS_INVALID_ID

[in] 마우스 포커스를 수신한 컨트롤의 식별자.

Return Value

컨트롤은 비활성화 결과를 제어합니다.

© 2000-2025, MetaQuotes Ltd.

Save

컨테이너 정보를 파일에 저장합니다.

```
virtual bool  Save(
   const int  file_handle      // 핸들
   )
```

Parameters

file_handle

[in] 쓰기 위해 이전에 열었던 이진 파일의 핸들.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Load

파일에서 컨테이너 정보를 로드합니다

```
virtual bool  Load(
   const int  file_handle      // 핸들
   )
```

Parameters

file_handle

[in] 읽기 위해 이전에 연 이진 파일의 핸들.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnResize

컨트롤 "Resize" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnActivate

컨트롤의 가상 핸들러가 내부 이벤트를 "활성화"합니다.

```
virtual bool  OnActivate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

컨트롤 "Deactivate" 내부 이벤트의 가상 핸들러

```
virtual bool  OnDeactivate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

CLabel

CLabelは "Text label" 차트 개체에 기반한 단순 컨트롤의 클래스입니다.

Description

CLabelは 편집 불가능한 단순한 텍스트 레이블을 만들기 위한 것입니다.

Declaration

```
class CLabel : public CWndObj
```

Title

```
#include <Controls\Label.mqh>
```

상속 계층

CObject
CWnd
CWndObj
CLabel

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
속성 변경 이벤트 핸들러	
OnSetText	"SetText" 이벤트 핸들러
OnSetColor	"SetColor" 이벤트 핸들러
OnSetFont	"SetFont" 이벤트 핸들러
OnSetFontSize	"SetFontSize" 이벤트 핸들러
내부 이벤트 핸들러	
OnCreate	내부 이벤트 핸들러 "생성"
OnShow	내부 이벤트 핸들러 "Show"
OnHide	내부 이벤트 핸들러 "Hide"
OnMove	내부 이벤트 핸들러 "Move"

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CWnd에서 상속된 메서드

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndObj에서 상속된 메서드

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

텍스트 레이블이 있는 패널 작성 예제:

```
//+------------------------------------------------------------------+
//|                                                ControlsLabel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CLabel"
#include <Controls\Dialog.mqh>
#include <Controls\Label.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                          |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CLabel            m_label;                        // CLabel 객체
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- 종속 컨트롤 생성
   bool              CreateLabel(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnClickLabel(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)

EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateLabel())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "CLabel"을 생성                                                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateLabel(void)
  {
//--- 좌표
   int x1=INDENT_RIGHT;
   int y1=INDENT_TOP+CONTROLS_GAP_Y;
   int x2=x1+100;
   int y2=y1+20;
//--- 생성
   if(!m_label.Create(m_chart_id,m_name+"Label",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_label.Text("Label"))
      return(false);
   if(!Add(m_label))
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

새 CLabel 컨트롤 생성.

```
virtual bool  Create(
   const long     chart,       // 차트 ID
   const string   name,        // 이름
   const int      subwin,      // 차트 하위창
   const int      x1,          // x1 좌표
   const int      y1,          // y1 좌표
   const int      x2,          // x2 좌표
   const int      y2           // y2 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetText

컨트롤 "SetText" (OBJPROP_TEXT 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetText()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

컨트롤 "SetColor" (OBJPROP_COLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColor()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

컨트롤 "SetFont" (OBJPROP_FONT 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetFont()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

컨트롤 "SetFontSize" (OBJPROP_FONTSIZE 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetFontSize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

컨트롤 "Create" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnCreate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnShow

컨트롤 "Show" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnHide

컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMove

컨트롤 "Move" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnMove()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CBmpButton

CBMPButton은 "비트맵 레이블" 차트 개체에 기반한 단순 컨트롤 클래스입니다.

Description

CBMPButton은 그래픽 이미지의 버튼을 만들기 위한 것입니다.

Declaration

```
class CBmpButton : public CWndObj
```

Title

```
#include <Controls\BmpButton.mqh>
```

상속 계층

CObject
CWnd
CWndObj
CBmpButton

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
Properties	
Border	컨트롤의 "경계" 속성을 가져오거나 설정합니다
BmpNames	컨트롤의 bmp 파일 이름을 설정합니다
BmpOffName	OFF 상태의 bmp 파일 이름을 가져오거나 설정합니다
BmpOnName	ON 상태의 bmp 파일 이름을 가져오거나 설정합니다
BmpPassiveName	수동 상태의 bmp 파일 이름을 가져오거나 설정합니다
BmpActiveName	활성 상태의 bmp 파일 이름을 가져오거나 설정합니다
State	
Pressed	컨트롤 상태를 가져오거나 설정합니다
Locking	컨트롤의 "잠금" 속성을 가져오거나 설정합니다
내부 이벤트 핸들러	
OnSetZOrder	내부 이벤트 핸들러 "SetZORDER"
OnCreate	내부 이벤트 핸들러 "생성"
OnShow	내부 이벤트 핸들러 "Show"
OnHide	내부 이벤트 핸들러 "Hide"
OnMove	내부 이벤트 핸들러 "Move"
OnChange	내부 이벤트 핸들러 "Change"
OnActivate	내부 이벤트 핸들러 "Activate"
OnDeactivate	내부 이벤트 핸들러 "Deactivate"
OnMouseDown	내부 이벤트 핸들러 "MouseDown"
OnMouseUp	내부 이벤트 핸들러 "MouseUp"

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CWnd에서 상속된 메서드

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndObj에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

비트맵 레이블을 사용하여 패널을 만드는 예:

```
//+------------------------------------------------------------------+
//|                                              ControlsBmpButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CBmpButton"
#include <Controls\Dialog.mqh>
#include <Controls\BmpButton.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CBmpButton        m_bmpbutton1;                  // CBmpButton 객체
   CBmpButton        m_bmpbutton2;                  // CBmpButton 객체
```

© 2000-2025, MetaQuotes Ltd.

```
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- 종속 컨트롤 생성
   bool              CreateBmpButton1(void);
   bool              CreateBmpButton2(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnClickBmpButton1(void);
   void              OnClickBmpButton2(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_bmpbutton1,OnClickBmpButton1)
ON_EVENT(ON_CLICK,m_bmpbutton2,OnClickBmpButton2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateBmpButton1())
      return(false);
   if(!CreateBmpButton2())
      return(false);
//--- 성공
   return(true);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| "BmpButton1" 버튼 생성                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton1(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- 생성
   if(!m_bmpbutton1.Create(m_chart_id,m_name+"BmpButton1",m_subwin,x1,y1,x2,y2))
      return(false);
//--- 컨트롤 CBmpButton의 bmp 파일 이름을 설정합니다
   m_bmpbutton1.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton1))
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "BmpButton2" 고정 버튼 만들기                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton2(void)
  {
//--- 좌표
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- 생성
   if(!m_bmpbutton2.Create(m_chart_id,m_name+"BmpButton2",m_subwin,x1,y1,x2,y2))
      return(false);
//--- 컨트롤 CBmpButton의 bmp 파일 이름을 설정합니다
   m_bmpbutton2.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton2))
      return(false);
   m_bmpbutton2.Locking(true);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton1(void)
  {
   Comment(__FUNCTION__);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton2(void)
  {
   if(m_bmpbutton2.Pressed())
      Comment(__FUNCTION__+" 컨트롤 상태: On");
   else
      Comment(__FUNCTION__+" 컨트롤 상태: Off");
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Create

새 CBmpButton 컨트롤을 만듭니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // x1 좌표
   const int      y1,         // y1 좌표
   const int      x2,         // x2 좌표
   const int      y2          // y2 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Border (Get method)

컨트롤의 "Border"(경계 너비) 속성을 가져옵니다.

```
int  Border()  const
```

Return Value

"Border" 속성.

Border (Set method)

컨트롤의 "Border"(경계 너비) 속성을 설정합니다.

```
bool  Border(
   const int  value      // 값
   )
```

Parameters

value

[in] "Border" 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

BmpNames

컨트롤의 bmp 파일 이름을 설정합니다

```
bool  BmpNames(
   const string  off="",      // 파일명
   const string  on=""        // 파일명
   )
```

Parameters

off=""

[in] OFF 상태의 bmp 파일 이름.

on=""

[in] ON 상태의 bmp 파일 이름.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

BmpOffName (Get method)

OFF 상태의 bmp 파일명을 가져옵니다.

```
string  BmpOffName()  const
```

Return Value

OFF 상태의 bmp 파일 이름.

BmpOffName (Set method)

OFF 상태의 bmp 파일 이름을 설정합니다.

```
bool  BmpOffName(
   const string  name      // 파일명
   )
```

Parameters

name

[in] OFF 상태의 bmp 파일 이름.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

BmpOnName (Get method)

ON 상태의 bmp 파일 이름을 가져옵니다.

```
string  BmpOnName()  const
```

Return Value

ON 상태의 bmp 파일 이름.

BmpOnName (Set method)

ON 상태의 bmp 파일 이름을 설정합니다.

```
bool  BmpOnName(
   const string  name      // 파일명
   )
```

Parameters

name

[in] ON 상태의 bmp 파일 이름.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

BmpPassiveName (Get method)

제어 수동 상태에 대한 bmp 파일의 이름을 가져옵니다.

```
string  BmpPassiveName()  const
```

Return Value

제어 수동 상태의 bmp 파일 이름.

BmpPassiveName (Set method)

수동 상태의 bmp 파일 이름을 설정합니다.

```
bool  BmpPassiveName(
   const string  name      // 파일명
   )
```

Parameters

name

[in] 제어 수동 상태의 bmp 파일 이름.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

BmpActiveName (Get method)

활성 상태의 bmp 파일명을 가져옵니다.

```
string  BmpActiveName()  const
```

Return Value

활성 상태의 bmp 파일 이름.

Note

마우스 커서 그것 위에 떠 있는 제어 활발히 진행됩니다.

BmpActiveName (Set method)

활성 상태의 bmp 파일 이름을 설정합니다.

```
bool  BmpActiveName(
   const string  name      // 파일명
   )
```

Parameters

name

[in] 활성 상태의 bmp 파일명.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Pressed (Get method)

컨트롤의 상태("Pressed" 속성)를 가져옵니다.

```
bool  Pressed()  const
```

Return Value

컨트롤 상태.

Pressed (Set method)

컨트롤의 상태("Pressed" 속성)를 설정합니다.

```
bool  Pressed(
   const bool  pressed      // 상태
   )
```

Parameters

pressed

[in] 새 컨트롤 상태.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Locking (Get method)

컨트롤의 "Locking" 속성을 가져옵니다.

```
bool  Locking()  const
```

Return Value

"Locking" 속성 값.

Locking (Set method)

컨트롤의 "Locking" 속성의 새 값을 설정합니다.

```
void  Locking(
   const bool  locking      // 값
   )
```

Parameters

locking

[in] "Locking" 속성의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

컨트롤 "SetZOrder" (OBJPROP_ZORDER 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetZOrder()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

컨트롤 "Create" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnCreate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnShow

컨트롤 "Show" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnHide

컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMove

컨트롤 "Move" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnMove()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChange

컨트롤 "Change" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChange()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnActivate

컨트롤의 가상 핸들러가 내부 이벤트를 "활성화"합니다.

```
virtual bool  OnActivate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnDeactivate

컨트롤 "Deactivate" 내부 이벤트의 가상 핸들러

```
virtual bool  OnDeactivate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

컨트롤 "MouseDown" 내부 이벤트의 가상 핸들러

```
virtual bool  OnMouseDown()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

컨트롤 "MouseUp"(마우스 단추 해제) 이벤트의 가상 핸들러.

```
virtual bool  OnMouseUp()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CButton

CButton은 "Button" 차트 객체에 기반한 단순 제어의 클래스입니다.

Description

CButton 클래스는 단순 버튼을 만들기 위한 것입니다.

Declaration

```
class CButton : public CWndObj
```

Title

```
#include <Controls\Button.mqh>
```

상속 계층

CObject
CWnd
CWndObj
CButton

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
State	
Pressed	"Pressed" 속성을 가져오거나 설정합니다.
Locking	"Locking" 속성을 가져오거나 설정합니다.
속성 변경 이벤트 핸들러	
OnSetText	"SetText" 이벤트 핸들러
OnSetColor	"SetColor" 이벤트 핸들러
OnSetColorBackground	"SetColorBackground" 이벤트 핸들러
OnSetColorBorder	"SetColorBorder" 이벤트 핸들러
OnSetFont	"SetFont" 이벤트 핸들러
OnSetFontSize	"SetFontSize" 이벤트 핸들러
내부 이벤트 핸들러	
OnCreate	내부 이벤트 핸들러 "생성"
OnShow	내부 이벤트 핸들러 "Show"
OnHide	내부 이벤트 핸들러 "Hide"
OnMove	내부 이벤트 핸들러 "Move"
OnResize	"Resize" 내부 이벤트 핸들러
OnMouseDown	내부 이벤트 핸들러 "MouseDown"
OnOnMouseUp	내부 이벤트 핸들러 "MouseUp"

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CWnd에서 상속된 메서드

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndObj에서 상속된 메서드

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Example of creating a panel with button:

```
//+------------------------------------------------------------------+
//|                                                ControlsButton.mq5 |
```

© 2000-2025, MetaQuotes Ltd.

```
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CButton"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CButton           m_button1;                     // 버튼 객체
   CButton           m_button2;                     // 버튼 객체
   CButton           m_button3;                     // 고정 버튼 객체

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 종속 컨트롤 생성
   bool              CreateButton1(void);
   bool              CreateButton2(void);
   bool              CreateButton3(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnClickButton1(void);
   void              OnClickButton2(void);
   void              OnClickButton3(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_button1,OnClickButton1)
ON_EVENT(ON_CLICK,m_button2,OnClickButton2)
ON_EVENT(ON_CLICK,m_button3,OnClickButton3)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateButton1())
      return(false);
   if(!CreateButton2())
      return(false);
   if(!CreateButton3())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "Button1" 버튼 생성                                              |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
bool CControlsDialog::CreateButton1(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- 생성
   if(!m_button1.Create(m_chart_id,m_name+"Button1",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button1.Text("Button1"))
      return(false);
   if(!Add(m_button1))
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "Button2" 버튼 생성                                              |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton2(void)
  {
//--- 좌표
   int x1=INDENT_LEFT+(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- 생성
   if(!m_button2.Create(m_chart_id,m_name+"Button2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_button2.Text("Button2"))
      return(false);
   if(!Add(m_button2))
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "Button3" 고정 버튼 생성                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton3(void)
  {
//--- 좌표
   int x1=INDENT_LEFT+2*(BUTTON_WIDTH+CONTROLS_GAP_X);
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- 생성
   if(!m_button3.Create(m_chart_id,m_name+"Button3",m_subwin,x1,y1,x2,y2))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
   if(!m_button3.Text("Locked"))
      return(false);
   if(!Add(m_button3))
      return(false);
   m_button3.Locking(true);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton2(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton3(void)
  {
   if(m_button3.Pressed())
      Comment(__FUNCTION__+" 컨트롤 상태: On");
   else
      Comment(__FUNCTION__+" 컨트롤 상태: Off");
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

새 CButton 컨트롤 생성.

```
virtual bool  Create(
   const long     chart,       // 차트 ID
   const string   name,        // 이름
   const int      subwin,      // 차트 하위창
   const int      x1,          // 좌표
   const int      y1,          // 좌표
   const int      x2,          // 좌표
   const int      y2           // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Pressed (Get method)

컨트롤의 상태("Pressed" 속성)를 가져옵니다.

```
bool  Pressed()  const
```

Return Value

컨트롤의 "Pressed" 속성.

Pressed (Set method)

컨트롤의 상태("Pressed" 속성)를 설정합니다.

```
bool  Pressed(
   const bool  pressed      // 상태
   )
```

Parameters

pressed

[in] 새 컨트롤 상태.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Locking (Get method)

컨트롤의 "Locking" 속성을 가져옵니다.

```
bool  Locking()  const
```

Return Value

"Locking" 속성 값.

Locking (Set method)

컨트롤의 "Locking" 속성 값을 설정합니다.

```
void  Locking(
   const bool  locking      // 값
   )
```

Parameters

locking

[in] 컨트롤의 "Locking" 속성의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

OnSetText

컨트롤 "SetText" (OBJPROP_TEXT 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetText()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

컨트롤 "SetColor" (OBJPROP_COLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColor()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

컨트롤 "SetColorBackground" (OBJPROP_BGCOLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColorBackground()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

컨트롤 "SetColorBorder" (OBJPROP_BORDER_COLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColorBackground()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

컨트롤 "SetFont" (OBJPROP_FONT 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetFont()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

컨트롤 "SetFontSize" (OBJPROP_FONTSIZE 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetFontSize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

컨트롤 "Create" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnCreate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnShow

컨트롤 "Show" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnHide

컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMove

컨트롤 "Move" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnMove()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnResize

컨트롤 "Resize" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMouseDown

컨트롤 "MouseDown" 이벤트 핸들러의 가상 핸들러.

```
virtual bool  OnMouseDown()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

컨트롤에서 마우스 왼쪽 버튼을 누르면 "MouseDown" 이벤트가 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnMouseUp

컨트롤 "MouseUp"(마우스 왼쪽 버튼 해제) 이벤트의 가상 핸들러.

```
virtual bool  OnMouseUp()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

컨트롤 "MouseUp"(마우스 왼쪽 버튼 해제) 이벤트의 가상 핸들러.

© 2000-2025, MetaQuotes Ltd.

CEdit

CEdit은 "Edit" 차트 개체에 기반한 단순 컨트롤의 클래스입니다.

Description

사용자가 텍스트를 입력할 수 있는 컨트롤을 만들기 위한 의 CEdit 클래스.

Declaration

```
class CEdit : public CWndObj
```

Title

```
#include <Controls\Edit.mqh>
```

상속 계층

CObject
  CWnd
    CWndObj
      CEdit

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
Properties	
ReadOnly	"ReadOnly" 속성을 가져오거나 설정합니다
TextAlign	"TextAlign" 속성을 가져오거나 설정합니다
차트 객체 이벤트 핸들러	
OnObjectEndEdit	CHARTEVENT_OBJECT_ENDEDIT 이벤트 핸들러(가상)
속성 변경 이벤트 핸들러	
OnSetText	"SetText" 이벤트 핸들러
OnSetColor	"SetColor" 이벤트 핸들러
OnSetColorBackground	"SetColorBackground" 이벤트 핸들러
OnSetColorBorder	"SetColorBorder" 이벤트 핸들러
OnSetFont	"SetFont" 이벤트 핸들러
OnSetFontSize	"SetFontSize" 이벤트 핸들러
OnSetZOrder	"SetZOrder" 이벤트 핸들러
내부 이벤트 핸들러	
OnCreate	내부 이벤트 핸들러 "생성"
OnShow	내부 이벤트 핸들러 "Show"
OnHide	내부 이벤트 핸들러 "Hide"
OnMove	내부 이벤트 핸들러 "Move"
OnResize	"Resize" 내부 이벤트 핸들러
OnChange	내부 이벤트 핸들러 "Change"
OnClick	"Click" 내부 이벤트 핸들러

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CWnd에서 상속된 메서드

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndObj에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

편집 컨트롤을 사용한 패널 작성 예제:

```
//+------------------------------------------------------------------+
//|                                                 ControlsEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CEdit"
#include <Controls\Dialog.mqh>
#include <Controls\Edit.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CEdit             m_edit;                        // CEdit object

public:
```

© 2000-2025, MetaQuotes Ltd.

```
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러

protected:
   //--- 종속 컨트롤 생성
   bool              CreateEdit(void);
  };
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateEdit())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 표시 필드 생성                                                   |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateEdit(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=ClientAreaWidth()-INDENT_RIGHT;
   int y2=y1+EDIT_HEIGHT;
//--- 생성
   if(!m_edit.Create(m_chart_id,m_name+"Edit",m_subwin,x1,y1,x2,y2))
      return(false);
//--- 내용 편집을 허용
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!m_edit.ReadOnly(false))
      return(false);
   if(!Add(m_edit))
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

CEdit 컨트롤을 생성합니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ReadOnly (Get Method)

컨트롤의 "ReadOnly" 특성을 가져옵니다.

```
bool  ReadOnly()
```

Return Value

"ReadOnly" 속성 값.

ReadOnly (Set Method)

컨트롤의 "ReadOnly" 속성 값을 설정합니다.

```
bool  ReadOnly(
   const bool  flag      // 값
   )
```

Parameters

flag

[in] "ReadOnly" 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

TextAlign (Get method)

컨트롤의 "TextAlign" 속성(텍스트 정렬 모드) 값을 가져옵니다.

```
ENUM_ALIGN_MODE  TextAlign() const
```

Return Value

컨트롤의 "TextAlign" 속성 값.

TextAlign (Set method)

컨트롤의 "TextAlign" 속성(텍스트 정렬 모드)의 새 값을 설정합니다.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // 속성 값
   )
```

Parameters

align

[in] "TextAlign" 속성의 새 값.

Return Value

성공시 true, 속성을 변경할 수 없으면 false.

© 2000-2025, MetaQuotes Ltd.

OnObjectEndEdit

CHARTEVENT_OBJECT_ENDEDIT 차트 객체 이벤트의 가상 핸들러.

```
virtual bool  OnObjectEndEdit()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetText

컨트롤 "SetText" (OBJPROP_TEXT 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetText()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColor

컨트롤 "SetColor" (OBJPROP_COLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColor()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

컨트롤 "SetColorBackground" (OBJPROP_BGCOLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColorBackground()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

컨트롤 "SetColorBorder" (OBJPROP_BORDER_COLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColorBackground()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetFont

컨트롤 "SetFont" (OBJPROP_FONT 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetFont()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetFontSize

컨트롤 "SetFontSize" (OBJPROP_FONTSIZE 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetFontSize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetZOrder

컨트롤 "SetZOrder" (OBJPROP_ZORDER 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetZOrder()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

컨트롤 "Create" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnCreate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnShow

컨트롤 "Show" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnHide

컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMove

컨트롤 "Move" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnMove()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnResize

컨트롤 "Resize" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChange

컨트롤 "Change" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChange()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClick

컨트롤의 "클릭"(마우스 단추 클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClick()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CPanel

CPanel은 "직각 레이블" 차트 개체에 기반한 단순 컨트롤 클래스입니다.

Description

CPanel 클래스는 제어 장치를 그룹에서 유사한 기능과 결합하기 위한 것입니다.

Declaration

```
class CPanel : public CWndObj
```

Title

```
#include <Controls\Panel.mqh>
```

상속 계층

CObject
CWnd
CWndObj
CPanel

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
차트 객체 속성	
BorderType	차트 객체의 "BorderType" 특성을 가져옵니다
차트 객체 이벤트 핸들러	
OnSetText	"SetText" 이벤트 핸들러
OnSetColorBackground	"SetColorBackground" 이벤트 핸들러
OnSetColorBorder	"SetColorBorder" 이벤트 핸들러
내부 이벤트 핸들러	
OnCreate	내부 이벤트 핸들러 "생성"
OnShow	내부 이벤트 핸들러 "Show"
OnHide	내부 이벤트 핸들러 "Hide"
OnMove	내부 이벤트 핸들러 "Move"
OnResize	"Resize" 내부 이벤트 핸들러
OnChange	내부 이벤트 핸들러 "Change"

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CWnd에서 상속된 메서드

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndObj에서 상속된 메서드

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

직사각형 레이블이 있는 패널 작성 예제:

```
//+------------------------------------------------------------------+
//|                                                ControlsPanel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CPanel"
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Controls\Dialog.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const

protected:
   //--- 종속 컨트롤 생성
   bool              CreatePanel(void);
  };
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreatePanel())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "CPanel" 생성                                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePanel(void)
  {
//--- 좌표
   int x1=20;
   int y1=20;
   int x2=ExtDialog.Width()/3;
   int y2=ExtDialog.Height()/3;
//--- 생성
   if(!my_white_border.Create(0,ExtDialog.Name()+"MyWhiteBorder",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!my_white_border.ColorBackground(CONTROLS_DIALOG_COLOR_BG))
      return(false);
   if(!my_white_border.ColorBorder(CONTROLS_DIALOG_COLOR_BORDER_LIGHT))
      return(false);
   if(!ExtDialog.Add(my_white_border))
      return(false);
   my_white_border.Alignment(WND_ALIGN_CLIENT,0,0,0,0);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//---
CPanel   my_white_border;        // object CPanel
bool     pause=true;             // true - pause
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   EventSetTimer(3);
   pause=true;
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------+
//|  Timer                                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
   pause=!pause;
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

새 CPanel 컨트롤을 생성합니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

BorderType (Get method)

차트 객체의 "BorderType" 특성을 가져옵니다.

```
ENUM_BORDER_TYPE  BorderType()
```

Return Value

"BorderType" 속성의 값.

BorderType (Set method)

차트 객체의 "BorderType" 속성의 새 값을 설정합니다.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // 값
   )
```

Parameters

type

[in] "BorderType" 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetText

컨트롤 "SetText" (OBJPROP_TEXT 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetText()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBackground

컨트롤 "SetColorBackground" (OBJPROP_BGCOLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColorBackground()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnSetColorBorder

컨트롤 "SetColorBorder" (OBJPROP_BORDER_COLOR 속성 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnSetColorBackground()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

컨트롤 "Create" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnCreate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnShow

컨트롤 "Show" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnHide

컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMove

컨트롤 "Move" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnMove()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnResize

컨트롤 "Resize" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChange

컨트롤 "Change" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChange()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CPicture

CPicture는 "비트맵 레이블" 객체를 기반으로 하는 단순 컨트롤의 클래스입니다.

Description

CPicture 클래스는 간단한 그래픽 이미지를 만들기 위한 것입니다.

Declaration

```
class CPicture : public CWndObj
```

Title

```
#include <Controls\Picture.mqh>
```

상속 계층

CObject
CWnd
CWndObj
CPicture

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
차트 객체 속성	
Border	차트 객체의 테두리 너비를 가져오거나 설정합니다
BmpName	컨트롤의 bmp 파일 이름을 가져오거나 설정합니다
내부 이벤트	
OnCreate	내부 이벤트 핸들러 "생성"
OnShow	내부 이벤트 핸들러 "Show"
OnHide	내부 이벤트 핸들러 "Hide"
OnMove	내부 이벤트 핸들러 "Move"
OnChange	내부 이벤트 핸들러 "Change"

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Save, Load, Type, Compare

클래스 CWnd에서 상속된 메서드

Destroy, OnMouseEvent, Name, ControlsTotal, Control, ControlFind, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Move, Move, Shift, Contains, Contains, Alignment, Align, Id, Id, IsEnabled, Enable, Disable, IsVisible, Visible, Show, Hide, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndObj에서 상속된 메서드

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

비트맵 레이블을 사용하여 패널을 만드는 예:

```
//+------------------------------------------------------------------+
//|                                                ControlsPicture.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CPicture"
#include <Controls\Dialog.mqh>
#include <Controls\Picture.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
```

© 2000-2025, MetaQuotes Ltd.

```
#define INDENT_LEFT                     (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                      (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                    (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                   (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                  (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                  (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                    (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                   (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                     (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                     (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                     (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                    (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                    (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                          |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CPicture          m_picture;                    // CPicture object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- 종속 컨트롤 생성
   bool              CreatePicture(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnClickPicture(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_picture,OnClickPicture)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreatePicture())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "Picture"를 생성합니다                                           |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePicture(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+32;
   int y2=y1+32;
//--- 생성
   if(!m_picture.Create(m_chart_id,m_name+"Picture",m_subwin,x1,y1,x2,y2))
      return(false);
//--- CPicture 컨트롤을 표시할 bmp 파일의 이름 설정
   m_picture.BmpName("\\Images\\euro.bmp");

   if(!Add(m_picture))
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickPicture(void)
  {
   Comment(__FUNCTION__);
  }
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

새 CPicture 컨트롤을 만듭니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Border (Get method)

컨트롤의 "Border"(경계 너비) 속성을 가져옵니다.

```
int  Border()  const
```

Return Value

"Border" 속성.

Border (Set method)

컨트롤의 "Border"(경계 너비) 속성을 설정합니다.

```
bool  Border(
   const int  value      // 값
   )
```

Parameters

value

[in] "Border" 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

BmpName (Get method)

컨트롤의 bmp 파일 이름을 가져옵니다.

```
string  BmpName()  const
```

Return Value

컨트롤의 bmp 파일 이름.

BmpName (Set method)

컨트롤의 bmp 파일 이름을 설정합니다.

```
bool  BmpName(
   const string  name      // 파일명
   )
```

Parameters

name

[in] 컨트롤의 bmp 파일 이름.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnCreate

컨트롤 "Create" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnCreate()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnShow

컨트롤 "Show" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnHide

컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnMove

컨트롤 "Move" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnMove()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChange

컨트롤 "Change" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChange()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CScroll

CScrollは スクロール 막대를 만드는 기본 클래스입니다.

Description

CScrollは (종속적인 컨트롤이 있는) 복잡한 컨트롤로 스크롤 바를 만들기 위한 기본 기능을 포함하고 있습니다. 기본 클래스 자체는 별도의 컨트롤로 사용되지 않으며 상속자 중 두 개(CScrollV 및 CScrollH 클래스)가 컨트롤로 사용됩니다.

Declaration

```
class CScroll : public CWndContainer
```

Title

```
#include <Controls\Scrolls.mqh>
```

상속 계층

CObject
CWnd
CWndContainer
CScroll

직계 종속

CScrollH, CScrollV

그룹별 클래스 메서드

Create	
Create	컨트롤 생성
차트 객체 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Properties	
MinPos	최소 위치를 가져오거나 설정합니다
MaxPos	최대 위치를 가져오거나 설정합니다
CurrPos	현재 위치를 가져오거나 설정합니다
종속 컨트롤 생성	
CreateBack	배경 버튼을 작성합니다
CreateInc	스크롤 막대의 감소 버튼을 생성합니다
CreateDec	스크롤 막대의 감소 버튼을 작성합니다
CreateThumb	스크롤 막대의 엄지 버튼(끌 수 있음)를 작성합니다

© 2000-2025, MetaQuotes Ltd.

Create	
종속 컨트롤 이벤트 핸들러	
OnClickInc	증가 버튼 이벤트를 처리하는 데 사용되는 이벤트 핸들러
OnClickDec	감소 버튼 이벤트를 처리하는 데 사용되는 이벤트 핸들러
내부 이벤트 핸들러	
OnShow	내부 이벤트 핸들러 "생성"
OnHide	내부 이벤트 핸들러 "Hide"
OnChangePos	"ChangePosition" 내부 이벤트 핸들러
객체 드래그 핸들러	
OnThumbDragStart	"ThumbDragStart" 이벤트 핸들러
OnThumbDragProcess	"ThumbDragProcess" 이벤트 핸들러
OnThumbDragEnd	"ThumbDragEnd" 이벤트 핸들러
Position	
CalcPos	좌표별 스크롤 막대 위치를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000-2025, MetaQuotes Ltd.

Create

새 CScroll 컨트롤을 생성합니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

MinPos (Get method)

CScroll 컨트롤의 "MinPos" (minimal position) 값을 가져옵니다.

```
int  MinPos()  const
```

Return Value

"MinPos" 속성의 새 값.

MinPos (Set method)

CScroll 컨트롤의 "MinPos"(최소 포지션) 값을 설정합니다.

```
void  MinPos(
   const int  value      // 값
   )
```

Parameters

value

[in] "MinPos" 속성의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

MaxPos (Get method)

CScroll 컨트롤의 "MaxPos"(최대 포지션) 값을 가져옵니다.

```
int  MaxPos()  const
```

Return Value

"MaxPos" 속성의 새 값.

MaxPos (Set method)

CScroll 컨트롤의 "MaxPos"(최대 포지션) 값을 설정합니다.

```
void  MaxPos(
   const int  value      // 값
   )
```

Parameters

value

[in] "MaxPos" 속성의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

CurrPos (Get method)

CScroll 컨트롤의 "CurrPos"(현재 포지션) 값을 가져옵니다.

```
int  CurrPos()  const
```

Return Value

"CurrPos" 속성의 새 값.

CurrPos (Set method)

CScroll 컨트롤의 "CurrPos"(현재 포지션) 값을 설정합니다.

```
void  CurrPos(
   const int  value      // 값
   )
```

Parameters

value

[in] "CurrPos" 속성의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

CreateBack

CScroll 컨트롤 배경 버튼을 생성합니다.

```
virtual bool  CreateBack()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateInc

CScroll 컨트롤 상승 버튼을 생성합니다.

```
virtual bool  CreateInc()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateDec

CScroll 컨트롤 하강 버튼을 생성합니다.

```
virtual bool  CreateDec()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

CScroll 컨트롤의 엄지 버튼(드래그 가능)을 생성합니다.

```
virtual bool  CreateThumb()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickInc

컨트롤 "ClickInc"(증가 버튼에 마우스 좌클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickInc()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickDec

컨트롤 "ClickDec" (하강 버튼에 마우스 좌클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickDec()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnShow

컨트롤 "Show" 이벤트의 가상 핸들러.

```
virtual bool  OnShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnHide

컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

컨트롤 "ChangePos"(포지션 변경) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChangePos()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

컨트롤 "ThumbDragStart"(드래그 시작) 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragStart()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragStart" 이벤트는 드래그 작업의 시작에 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

컨트롤 "ThumbDragProcess" 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x 좌표
   const int  y       // y 좌표
   )
```

Parameters

x

[in] 마우스 커서의 현재 X 좌표.

y

[in] 마우스 커서의 현재 Y 좌표.

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragProcess"는 스크롤 바 컨트롤(엄지 버튼)이 움직일 때 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

컨트롤 "ThumbDragEnd"(드래그 절차 완료) 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragEnd()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragEnd"는 스크롤 바 컨트롤(엄지 버튼)의 드래그 작업을 마칠때 발생합니다.

© 2000-2025, MetaQuotes Ltd.

CalcPos

컨트롤 스크롤 바의 위치를 좌표로 가져옵니다.

```
virtual int  CalcPos(
   const int      coord      // 좌표
   )
```

Parameters

coord

[in] 스크롤 바 좌표.

Return Value

스크롤 바 위치.

© 2000-2025, MetaQuotes Ltd.

CScrollV

CScrollV은 "수직 스크롤 바" 복합 컨트롤 클래스입니다.

Description

CScrollV 클래스는 수직 스크롤 막대를 만들기 위한 것입니다.

Declaration

```
class CScrollV : public CScroll
```

Title

```
#include <Controls\Scrolls.mqh>
```

상속 계층

CObject
CWnd
CWndContainer
CScroll
CScrollV

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

© 2000-2025, MetaQuotes Ltd.

종속 제어	
CreateInc	스크롤바 증가 버튼을 생성합니다
CreateDec	스크롤바 감소 버튼을 생성합니다
CreateThumb	스크롤바 엄지 버튼(드래그 가능)을 생성합니다
내부 이벤트 핸들러	
OnResize	"Resize" 내부 이벤트 핸들러
OnChangePos	"ChangePosition" 내부 이벤트 핸들러
이벤트 핸들러 드래그	
OnThumbDragStart	"ThumbDragStart" 이벤트 핸들러
OnThumbDragProcess	"ThumbDragProcess" 이벤트 핸들러
OnThumbDragEnd	"ThumbDragEnd" 이벤트 핸들러
Position	
CalcPos	좌표별 스크롤 막대 위치를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

클래스 CScroll로부터 상속된 메서드

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

수직 스크롤 막대를 사용한 패널 작성 예제:

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CScrollV"
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollV          m_scroll_v;                    // CScrollV object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- 종속 컨트롤 생성
   bool              CreateScrollV(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateScrollV())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| CScrollsV 객체를 생성합니다                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollV(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+18;
   int y2=y1+LIST_HEIGHT;
//--- 생성
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollV",m_subwin,x1,y1,x2,y2))
      return(false);
//--- 스크롤바를 설정합니다
   m_scroll_v.MinPos(0);
//--- 스크롤바를 설정합니다
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // 이벤트 ID
                  const long& lparam,    // long 타입의 이벤트 매개 변수
```

© 2000-2025, MetaQuotes Ltd.

```
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

© 2000-2025, MetaQuotes Ltd.

CreateInc

컨트롤의 증가 버튼을 생성합니다.

```
virtual bool  CreateInc()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateDec

컨트롤의 감소 버튼을 생성합니다.

```
virtual bool  CreateDec()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

컨트롤의 엄지 버튼(드래그 가능)을 생성합니다.

```
virtual bool  CreateThumb()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnResize

"Resize" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

컨트롤 "ChangePos"(포지션 변경) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChangePos()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

컨트롤 "ThumbDragStart"(드래그 시작) 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragStart()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragStart" 이벤트는 드래그 작업의 시작에 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

컨트롤 "ThumbDragProcess" 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x 좌표
   const int  y       // y 좌표
   )
```

Parameters

x

[in] 마우스 커서의 현재 X 좌표.

y

[in] 마우스 커서의 현재 Y 좌표.

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragProcess"는 스크롤 바 컨트롤(엄지 버튼)이 움직일 때 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

컨트롤 "ThumbDragEnd"(드래그 절차 완료) 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragEnd()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragEnd"는 스크롤 바 컨트롤(엄지 버튼)의 드래그 작업을 마칠때 발생합니다.

© 2000-2025, MetaQuotes Ltd.

CalcPos

컨트롤 스크롤 바의 위치를 좌표로 가져옵니다.

```
virtual int  CalcPos(
   const int      coord      // 좌표
   )
```

Parameters

coord

[in] 스크롤 바 좌표.

Return Value

스크롤 바 위치.

© 2000-2025, MetaQuotes Ltd.

CScrollH

CScrollH은 "수평 스크롤 바" 복합 컨트롤 클래스입니다.

Description

CScrollH은 수평 스크롤 바를 생성합니다.

Declaration

```
class CScrollH : public CScroll
```

Title

```
#include <Controls\Scrolls.mqh>
```

상속 계층

COBject
CWnd
CWndContainer
CScroll
CScrollH

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

© 2000-2025, MetaQuotes Ltd.

종속 제어	
CreateInc	스크롤바 증가 버튼을 생성합니다
CreateDec	스크롤바 감소 버튼을 생성합니다
CreateThumb	스크롤바 엄지 버튼(드래그 가능)을 생성합니다
내부 이벤트 핸들러	
OnResize	"Resize" 내부 이벤트 핸들러
OnChangePos	"ChangePosition" 내부 이벤트 핸들러
이벤트 핸들러 드래그	
OnThumbDragStart	"ThumbDragStart" 이벤트 핸들러
OnThumbDragProcess	"ThumbDragProcess" 이벤트 핸들러
OnThumbDragEnd	"ThumbDragEnd" 이벤트 핸들러
Position	
CalcPos	좌표별 스크롤 막대 위치를 가져옵니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

클래스 CScroll로부터 상속된 메서드

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

수평 스크롤바와 패널을 생성하는 예:

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollH.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CScrollH"
```

© 2000-2025, MetaQuotes Ltd.

```
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollH          m_scroll_v;                    // CScrollH object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- 종속 컨트롤 생성
   bool              CreateScrollsH(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_SCROLL_INC,m_scroll_v,OnScrollInc)
ON_EVENT(ON_SCROLL_DEC,m_scroll_v,OnScrollDec)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateScrollsH())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| CScrollsH 객체를 생성합니다                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollsH(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+3*BUTTON_WIDTH;
   int y2=y1+18;
//--- 생성
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollsH",m_subwin,x1,y1,x2,y2))
      return(false);
//--- 스크롤바를 설정합니다
   m_scroll_v.MinPos(0);
//--- 스크롤바를 설정합니다
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
```

© 2000-2025, MetaQuotes Ltd.

```
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

© 2000-2025, MetaQuotes Ltd.

CreateInc

컨트롤의 증가 버튼을 생성합니다.

```
virtual bool  CreateInc()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateDec

컨트롤의 감소 버튼을 생성합니다.

```
virtual bool  CreateDec()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateThumb

컨트롤의 엄지 버튼(드래그 가능)을 생성합니다.

```
virtual bool  CreateThumb()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnResize

"Resize" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChangePos

컨트롤 "ChangePos"(포지션 변경) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChangePos()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragStart

컨트롤 "ThumbDragStart"(드래그 시작) 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragStart()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragStart" 이벤트는 드래그 작업의 시작에 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragProcess

컨트롤 "ThumbDragProcess" 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x 좌표
   const int  y       // y 좌표
   )
```

Parameters

x

[in] 마우스 커서의 현재 X 좌표.

y

[in] 마우스 커서의 현재 Y 좌표.

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragProcess"는 스크롤 바 컨트롤(엄지 버튼)이 움직일 때 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnThumbDragEnd

컨트롤 "ThumbDragEnd"(드래그 절차 완료) 이벤트의 가상 핸들러.

```
virtual bool  OnThumbDragEnd()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"ThumbDragEnd"는 스크롤 바 컨트롤(엄지 버튼)의 드래그 작업을 마칠때 발생합니다.

© 2000-2025, MetaQuotes Ltd.

CalcPos

컨트롤 스크롤 바의 위치를 좌표로 가져옵니다.

```
virtual int  CalcPos(
   const int      coord      // 좌표
   )
```

Parameters

coord

[in] 스크롤 바 좌표.

Return Value

스크롤 바 위치.

© 2000-2025, MetaQuotes Ltd.

CWndClient

CWndClient는 (종속 제어가 있는) "클라이언트 영역" 복합 제어의 한 클래스입니다. 스크롤 막대 영역을 만들기 위한 기본 클래스입니다.

Description

CWndClient는 스크롤 막대를 사용하여 클라이언트 영역을 만드는 기능을 구현합니다.

Declaration

```
class CWndClient : public CWndContainer
```

Title

```
#include <Controls\WndClient.mqh>
```

상속 계층

CObject
  CWnd
    CWndContainer
      CWndClient

직계 종속

CCheckGroup, CListView, CRadioGroup

그룹별 클래스 메서드

Create	
Create	컨트롤 생성
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Properties	
ColorBackground	배경색을 설정합니다
ColorBorder	테두리 색을 설정합니다
BorderType	테두리 유형을 설정합니다
Settings	
VScrolled	수직 스크롤 막대가 사용됨을 나타내는 플래그를 가져오거나 설정합니다
HScrolled	수평 스크롤 막대가 사용됨을 나타내는 플래그를 가져오거나 설정합니다
종속 제어	

© 2000-2025, MetaQuotes Ltd.

Create	
CreateBack	스크롤 막대의 배경을 생성합니다
CreateScrollV	수직 스크롤 막대를 생성합니다
CreateScrollH	수평 스크롤 막대를 생성합니다
내부 이벤트 핸들러	
OnResize	"Resize" 내부 이벤트 핸들러
종속 컨트롤 이벤트 핸들러	
OnVScrollShow	VScroll 종속 컨트롤의 "Show" 내부 이벤트 핸들러(가상)
OnVScrollHide	VScroll 종속 컨트롤의 "Hide" 내부 이벤트 핸들러(가상)
OnHScrollShow	HScroll 종속 컨트롤의 "Show" 내부 이벤트 핸들러(가상)
OnHScrollHide	HScroll 종속 컨트롤의 "Hide" 내부 이벤트 핸들러(가상)
OnScrollLineDown	VScroll 종속 컨트롤의 "ScrollLineDown" 내부 이벤트 핸들러(가상)
OnScrollLineUp	VScroll 종속 컨트롤의 "ScrollLineUp" 내부 이벤트 핸들러(가상)
OnScrollLineLeft	HScroll 종속 컨트롤의 "ScrollLineLeft" 내부 이벤트 핸들러(가상)
OnScrollLineRight	HScroll 종속 컨트롤의 "ScrollLineRight" 내부 이벤트 핸들러(가상)
Resize	
Rebound	CRect 클래스 좌표를 사용하여 컨트롤의 새 매개변수를 설정합니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

© 2000-2025, MetaQuotes Ltd.

Create

새 CWndClient 컨트롤을 작성합니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ColorBackground

컨트롤의 배경색을 설정합니다.

```
bool  ColorBackground(
   const color  value      // 배경색
   )
```

Parameters

value

[in] 컨트롤의 배경색.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ColorBorder

컨트롤의 테두리 색상을 설정합니다.

```
bool  ColorBorder(
   const color  value      // 색상
   )
```

Parameters

value

[in] 컨트롤의 테두리 색상.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

BorderType

컨트롤의 테두리 유형을 설정합니다.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // 값
   )
```

Parameters

type

[in] 컨트롤의 테두리 유형.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

VScrolled (Get method)

수직 스크롤 막대가 사용되었음을 나타내는 플래그를 가져옵니다.

```
bool  VScrolled()
```

Return Value

수직 스크롤 막대가 사용되면 true, 그렇지 않으면 false.

VScrolled (Set method)

수직 스크롤 막대가 사용되었음을 나타내는 플래그를 설정합니다.

```
bool  VScrolled(
   const bool  flag      // 플래그
   )
```

Parameters

flag

[in] 플래그.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

HScrolled (Get method)

수평 스크롤 막대가 사용되었음을 나타내는 플래그를 가져옵니다.

```
bool  HScrolled()
```

Return Value

수평 스크롤이 사용되었으면 true, 그렇지 않으면 false.

HScrolled (Set method)

수평 스크롤 막대가 사용되었음을 나타내는 플래그를 설정합니다.

```
bool  HScrolled(
   const bool  flag      // 플래그
   )
```

Parameters

flag

[in] 플래그.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateBack

컨트롤의 배경 버튼을 생성합니다.

```
virtual bool  CreateBack()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateScrollV

수직 스크롤 막대를 생성합니다.

```
virtual bool  CreateScrollV()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateScrollH

수평 스크롤 막대를 생성합니다.

```
virtual bool  CreateScrollH()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnResize

"Resize" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

VScroll(수직 스크롤) 종속 컨트롤 "Show"(수직 스크롤 막대 쇼) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnVScrollShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

VScroll(수직 스크롤) 종속 컨트롤 "Hide"(수직 스크롤 막대 숨기기) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnVScrollHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnHScrollShow

HScroll(수평 스크롤) 종속 컨트롤 "Show"(수평 스크롤 막대 쇼) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnHScrollShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnHScrollHide

HScroll(수평 스크롤) 종속 컨트롤의 가상 핸들러 "Hide"(수평 스크롤 막대 숨기기) 내부 이벤트입니다.

```
virtual bool  OnHScrollHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

VScroll(수직 스크롤) 종속적 컨트롤의 가상 핸들러는 "ScrollLineDown"(수직 스크롤 라인 아래로) 내부 이벤트 핸들러.

```
virtual bool  OnScrollLineDown()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

VScroll(수직 스크롤) 종속 컨트롤 "ScrollLineUp"(수직 스크롤 라인업) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnScrollLineUp()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineLeft

HScroll(수평 스크롤) 종속 컨트롤 "ScrollLineLeft"(수평 스크롤 라인 왼쪽) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnScrollLineLeft()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineRight

HScroll(수평 스크롤) 종속 컨트롤 "ScrollLineRight"(수평 스크롤 라인 오른쪽) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnScrollLineRight()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

기본 클래스 메서드가 항상 true를 반환하는 것은 아닙니다.

© 2000-2025, MetaQuotes Ltd.

ReBound

CRect 클래스 좌표를 사용하여 컨트롤의 새 매개변수를 설정합니다.

```
void  ReBound(
   const & CRect  rect      // CRect 클래스
   )
```

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

CListView

CListView는 ListView 복합 컨트롤의 클래스입니다(종속 컨트롤 포함).

Description

CListView 클래스는 목록 제어 기능을 캡슐화합니다.

Declaration

```
class CListView : public CWndClient
```

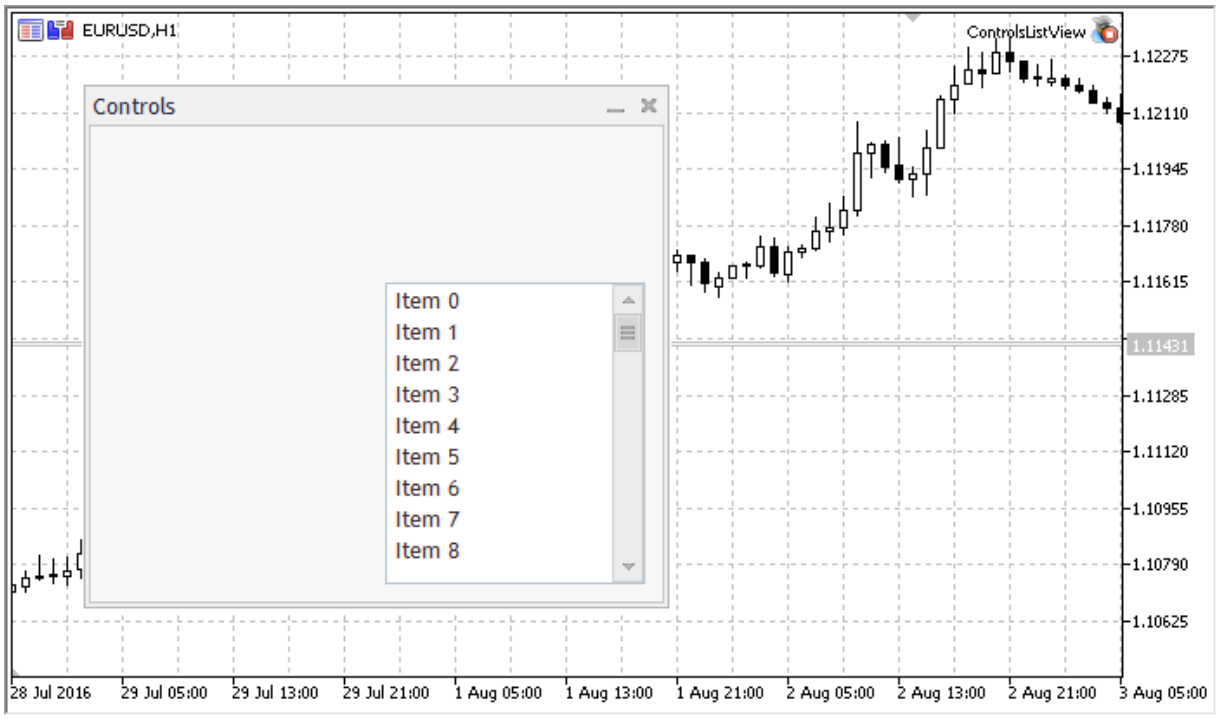
Title

```
#include <Controls\ListView.mqh>
```

상속 계층

CObject
CWnd
CWndContainer
CWndClient
CListView

아래에 제시된 코드의 결과:



그룹별 클래스 메서드

© 2000-2025, MetaQuotes Ltd.

Create	
Create	컨트롤 생성
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Settings	
TotalView	컨트롤에 표시되는 항목 수를 설정합니다
Add/Delete	
AddItem	제어 목록에 항목을 추가합니다
Data	
Select	인덱스별로 현재 목록 요소를 선택합니다
SelectByText	현재 목록 요소를 텍스트별로 선택합니다
SelectByValue	값별로 현재 목록 요소를 선택합니다
읽기전용 데이터	
Value	현재 목록 요소의 값을 가져옵니다
종속 제어	
CreateRow	ListView 행을 생성합니다
내부 이벤트 핸들러	
OnResize	"Resize" 내부 이벤트 핸들러(가상)
종속 컨트롤 이벤트 핸들러	
OnVScrollShow	VScroll 종속 컨트롤의 "Show" 내부 이벤트 핸들러(가상)
OnVScrollHide	VScroll 종속 컨트롤의 "Hide" 내부 이벤트 핸들러(가상)
OnScrollLineDown	VScroll 종속 컨트롤의 "ScrollLineDown" 내부 이벤트 핸들러(가상)
OnScrollLineUp	VScroll 종속 컨트롤의 "ScrollLineUp" 내부 이벤트 핸들러(가상)
OnItemClick	"ItemClick" 내부 이벤트 핸들러(가상)
Redraw	
Redraw	컨트롤 리드로우
RowState	지정된 행의 상태를 설정합니다
CheckView	지정된 행의 "가시성"을 확인합니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

CWndClient 클래스에서 상속된 메서드

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

리스트 뷰 컨트롤을 사용하여 패널 작성 예제:

```
//+------------------------------------------------------------------+
//|                                                ControlsListView.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CListView"
#include <Controls\Dialog.mqh>
#include <Controls\ListView.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
```

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                          |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CListView         m_list_view;                     // CListView object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- 종속 컨트롤 생성
   bool              CreateListView(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnChangeListView(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_list_view,OnChangeListView)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
```

© 2000-2025, MetaQuotes Ltd.

```
   if(!CreateListView())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "ListView" 요소를 생성합니다                                     |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateListView(void)
  {
//--- 좌표
   int x1=INDENT_LEFT+GROUP_WIDTH+2*CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+2*CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+LIST_HEIGHT-CONTROLS_GAP_Y;
//--- 생성
   if(!m_list_view.Create(m_chart_id,m_name+"ListView",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_list_view))
      return(false);
//--- 문자열로 채웁니다
   for(int i=0;i<16;i++)
      if(!m_list_view.AddItem("Item "+IntegerToString(i)))
         return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeListView(void)
  {
   Comment(__FUNCTION__+" \""+m_list_view.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
```

© 2000-2025, MetaQuotes Ltd.

```
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

CListView 컨트롤을 생성합니다.

```
virtual bool  Create(
   const long     chart,       // 차트 ID
   const string   name,        // 이름
   const int      subwin,      // 차트 하위창
   const int      x1,          // 좌표
   const int      y1,          // 좌표
   const int      x2,          // 좌표
   const int      y2           // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 작성되는 하위 창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

TotalView

컨트롤에 표시되는 항목 수를 설정합니다.

```
bool  TotalView(
   const int  value      // 표시된 항목 수
   )
```

Parameters

value

[in] 표시된 항목 수.

Return Value

성공시 true, 아니면 false.

Note

표시된 항목 수는 한 번에만 지정할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

AddItem

제어 목록에 항목을 추가합니다.

```
bool  AddItem(
   const string  item,     // 텍스트
   const long    value     // 값
   )
```

Parameters

item

[in] 텍스트.

value

[in] 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Select

인덱스별로 현재 목록 요소를 선택합니다.

```
bool  Select(
   const int  index     // 인덱스
   )
```

Parameters

index

[in] 항목 색인.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

SelectByText

현재 목록 요소를 텍스트별로 선택합니다.

```
bool  SelectByText(
   const string  text     // 텍스트
   )
```

Parameters

text

[in] 요소 텍스트.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

SelectByValue

값별로 현재 목록 요소를 선택합니다.

```
bool  SelectByValue(
   const long  value     // 값
   )
```

Parameters

value

[in] 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Value

현재 목록 요소의 값을 가져옵니다.

```
long  Value()
```

Return Value

현재 목록 요소의 값.

© 2000-2025, MetaQuotes Ltd.

CreateRow

CListView 컨트롤의 행을 작성합니다.

```
bool  CreateRow(
   const int  index     // 인덱스
   )
```

Parameters

index

[in] 항목 색인.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnResize

"Resize" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnResize()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

VScroll(수직 스크롤) 종속 컨트롤 "VScrollShow"(수직 스크롤 막대 쇼) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnVScrollShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

VScroll(수직 스크롤) 종속 컨트롤 "VscroellHide"(수직 스크롤 막대 숨기기) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnVScrollHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

VScroll(수직 스크롤) 종속적 컨트롤의 가상 핸들러는 "ScrollLineDown"(수직 스크롤 라인 아래로) 내부 이벤트 핸들러.

```
virtual bool  OnScrollLineDown()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

VScroll(수직 스크롤) 종속 컨트롤 "ScrollLineUp"(수직 스크롤 라인업) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnScrollLineUp()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnItemClick

CListView 컨트롤의 지정된 행에 있는 "ItemClick"(마우스 단추 클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnItemClick()
   const int    index     // 인덱스
   )
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Redraw

컨트롤 다시 그리기.

```
bool  Redraw()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

RowState

CListView 컨트롤의 지정된 행 상태를 변경합니다.

```
bool  RowState(
   const int   index      // 인덱스
   const bool  select     // 상태
   )
```

Parameters

index

[in] 행 인덱스.

select

[in] 행 상태.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CheckView

CListView 컨트롤의 지정된 행에 대한 "가시성"을 확인합니다.

```
bool  CheckView()
```

Return Value

선택 행이 표시되면 true, 그렇지 않으면 false.

© 2000-2025, MetaQuotes Ltd.

CComboBox

CComboBox는 종속 컨트롤이 있는 ComboBox 복합 컨트롤의 클래스입니다.

Description

ComboBox는 정적 제어와 결합된 선택용 목록 박스로 구성됩니다. 사용자가 컨트롤 옆에 있는 드롭다운 화살표를 선택할 때 컨트롤의 목록 상자 부분이 아래로 떨어질 수 있습니다

Declaration

```
class CComboBox : public CWndContainer
```

Title

```
#include <Controls\ComboBox.mqh>
```

상속 계층

CObject
CWnd
CWndContainer
CComboBox

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Add	
AddItem	컨트롤에 항목을 추가합니다
Settings	
ListViewItems	컨트롤에 표시되는 항목 수를 설정합니다
Data	
Select	인덱스별로 현재 목록 요소를 선택합니다
SelectByText	텍스트별로 현재 목록 요소를 선택합니다
SelectByValue	값별로 현재 목록 요소를 선택합니다
읽기전용 데이터	
Value	현재 목록 요소의 값을 가져옵니다
종속 제어	
CreateEdit	종속 컨트롤을 만듭니다(편집)
CreateButton	종속 컨트롤 생성 (버튼)
CreateList	종속 컨트롤을 만듭니다(목록 보기)
종속 컨트롤 이벤트 핸들러	
OnClickEdit	"ClickEdit" 이벤트 핸들러(가상).
OnClickButton	"ClickButton" 내부 이벤트 핸들러 (가상)
OnChangeList	"ChangeList" 내부 이벤트 핸들러(가상)
드롭다운 목록 표시/숨기기	
ListShow	항목 목록 보이기
ListHide	항목 목록 가리기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Hide

콤보 상자 컨트롤을 사용하여 패널 만들기 예제:

```
//+------------------------------------------------------------------+
//|                                                ControlsComboBox.mq5 |
//|                        Copyright 2015, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2015, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CComboBox"
#include <Controls\Dialog.mqh>
#include <Controls\ComboBox.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CComboBox         m_combo_box;;                  // CComboBox 객체

public:
```

© 2000-2025, MetaQuotes Ltd.

```
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- 종속 컨트롤 생성
   bool              CreateComboBox(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnChangeComboBox(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_combo_box,OnChangeComboBox)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateComboBox())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "ComboBox" 요소를 생성합니다                                     |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateComboBox(void)
  {
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- 생성
   if(!m_combo_box.Create(m_chart_id,m_name+"ComboBox",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_combo_box))
      return(false);
//--- 문자열로 채웁니다
   for(int i=0;i<16;i++)
      if(!m_combo_box.ItemAdd("Item "+IntegerToString(i)))
         return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeComboBox(void)
  {
   Comment(__FUNCTION__+" \""+m_combo_box.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
```

© 2000-2025, MetaQuotes Ltd.

```
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,          // 이벤트 ID
                  const long& lparam,    // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

CComboBox 컨트롤을 만듭니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 채팅의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

AddItem

컨트롤에 항목을 추가합니다.

```
bool  AddItem(
   const string  item,     // 텍스트
   const long    value     // 값
   )
```

Parameters

item

[in] 텍스트.

value=0

[in] 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ListViewItems

CComboBox 컨트롤의 목록 요소 수를 설정합니다.

```
void  ListViewItems(
   const int    value     // 요소 수
   )
```

Parameters

value

[in] 드롭다운 목록 요소의 수.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Select

인덱스별로 현재 목록 요소를 선택합니다.

```
bool  Select(
   const int  index     // 인덱스
   )
```

Parameters

index

[in] 항목 색인.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

SelectByText

지정된 텍스트를 기준으로 현재 목록 요소를 선택합니다.

```
bool  SelectByText(
   const string  text     // 텍스트
   )
```

Parameters

text

[in] 아이템 텍스트.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

SelectByValue

지정된 값별로 현재 목록 요소를 선택합니다.

```
bool  SelectByValue(
   const long  value     // 값
   )
```

Parameters

value

[in] 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Value

현재 목록 요소의 값을 가져옵니다.

```
long  Value()
```

Return Value

현재 목록 요소의 값.

© 2000-2025, MetaQuotes Ltd.

CreateEdit

컨트롤의 종속 컨트롤(편집)을 만듭니다.

```
virtual bool  CreateEdit()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateButton

종속 컨트롤(버튼)을 만듭니다.

```
virtual bool  CreateButton()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateList

종속 컨트롤(목록 보기)을 만듭니다.

```
virtual bool  CreateList()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickEdit

컨트롤 "ClickEdit"(편집에서 마우스 클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickEdit()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

"ClickButton"(클릭버튼 마우스 클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickButton()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChangeList

"ChangeList"(목록 변경) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChangeList()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ListShow

컨트롤의 드롭다운 목록을 표시합니다.

```
virtual bool  ListShow()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ListHide

컨트롤의 드롭다운 목록을 숨깁니다.

```
virtual bool  ListHide()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CCheckBox

CChekBox는 CheckBox 복합 컨트롤의 클래스입니다.

Description

CCheckBox 컨트롤은 사용자가 마우스를 클릭할 때 참 또는 거짓 조건을 선택할 수 있는 확인란을 표시합니다.

Declaration

```
class CCheckBox : public CWndContainer
```

Title

```
#include <Controls\CheckBox.mqh>
```

상속 계층

CObject
CWnd
CWndContainer
CCheckBox

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Properties	
Text	컨트롤과 관련된 텍스트 레이블을 가져오거나 설정
Color	컨트롤과 연결된 텍스트 레이블의 색상을 가져오거나 설정
State	
Checked	컨트롤 선택 여부를 나타내는 값을 가져오거나 설정
Data	
Value	컨트롤과 관련된 값을 가져오거나 설정
종속 제어	
CreateButton	종속 컨트롤 생성 (버튼)
CreateLabel	종속 컨트롤 생성 (레이블)
종속 컨트롤 이벤트 핸들러	
ClickButton	"ClickButton" 내부 이벤트 핸들러 (가상)
ClickLabel	"ClickLabel" 내부 이벤트 핸들러 (가상)

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

체크박스 컨트롤로 패널 만들기 예제:

```
//+------------------------------------------------------------------+
//|                                            ControlsCheckBox.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
```

© 2000-2025, MetaQuotes Ltd.

```
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CCheckBox"
#include <Controls\Dialog.mqh>
#include <Controls\CheckBox.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckBox         m_check_box1;                    // CCheckBox 객체
   CCheckBox         m_check_box2;                    // CCheckBox 객체
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- 종속 컨트롤 생성
   bool              CreateCheckBox1(void);
   bool              CreateCheckBox2(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnChangeCheckBox1(void);
   void              OnChangeCheckBox2(void);
  };
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_box1,OnChangeCheckBox1)
ON_EVENT(ON_CHANGE,m_check_box2,OnChangeCheckBox2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateCheckBox1())
      return(false);
   if(!CreateCheckBox2())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "CheckBox" 요소 생성                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox1(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- 생성
   if(!m_check_box1.Create(m_chart_id,m_name+"CheckBox1",m_subwin,x1,y1,x2,y2))
```

© 2000-2025, MetaQuotes Ltd.

```
      return(false);
   if(!m_check_box1.Text("CheckBox1"))
      return(false);
   if(!m_check_box1.Color(clrBlue))
      return(false);
   if(!Add(m_check_box1))
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "CheckBox" 요소 생성                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckBox2(void)
  {
//--- 좌표
   int x1=INDENT_LEFT+GROUP_WIDTH+CONTROLS_GAP_X;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- 생성
   if(!m_check_box2.Create(m_chart_id,m_name+"CheckBox2",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!m_check_box2.Text("CheckBox2"))
      return(false);
   if(!m_check_box2.Color(clrBlue))
      return(false);
   if(!Add(m_check_box2))
      return(false);
   m_check_box2.Checked(true);
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox1(void)
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box1.Checked()));
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckBox2(void)
```

© 2000-2025, MetaQuotes Ltd.

```
  {
   Comment(__FUNCTION__+" : Checked="+IntegerToString(m_check_box2.Checked()));
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

CCheckBox 컨트롤을 생성합니다.

```
virtual bool  Create(
   const long     chart,       // 차트 ID
   const string   name,        // 이름
   const int      subwin,      // 차트 하위창
   const int      x1,          // 좌표
   const int      y1,          // 좌표
   const int      x2,          // 좌표
   const int      y2           // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Text (Get method)

컨트롤과 연결된 레이블의 텍스트를 가져옵니다.

```
string  Text()
```

Return Value

레이블 텍스트.

Text (Set method)

컨트롤과 관련된 레이블의 텍스트를 설정합니다.

```
bool  Text(
   const string  value      // 텍스트
   )
```

Parameters

value

[in] 레이블의 새 텍스트.

Return Value

성공시 true, 아니면 false.

Note

레이블 텍스트는 차트 개체의 OBJPROP_TEXT(텍스트)를 설정하여 지정합니다.

© 2000-2025, MetaQuotes Ltd.

Color (Get method)

컨트롤과 연결된 레이블의 색상을 가져옵니다.

```
color  Color()  const
```

Return Value

레이블 색상.

Color (Set method)

컨트롤과 연결된 레이블의 색상을 설정합니다.

```
bool  Color(
   const color  value      // 색상
   )
```

Parameters

value

[in] 새 레이블 색상.

Return Value

성공시 true, 아니면 false.

Note

레이블 색상은 차트 개체의 OBJPROP_COLOR(색상)를 설정하여 지정합니다.

© 2000-2025, MetaQuotes Ltd.

Checked (Get method)

컨트롤 상태를 가져옵니다.

```
bool  Checked()  const
```

Return Value

컨트롤 상태.

Checked (Set method)

컨트롤 상태를 설정합니다.

```
bool  Checked(
   const bool  flag      // 상태
   )
```

Parameters

flag

[in] 새 상태.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Value (Get method)

컨트롤과 관련된 값을 가져옵니다.

```
int  Value()  const
```

Return Value

컨트롤과 관련된 값.

Value (Set method)

컨트롤과 관련된 값을 설정합니다.

```
void  Value(
   const int  value      // 값
   )
```

Parameters

value

[in] 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

CreateButton

종속 컨트롤(버튼)을 만듭니다.

```
virtual bool  CreateButton()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateLabel

종속 컨트롤(레이블)을 생성합니다.

```
virtual bool  CreateLabel()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

컨트롤 "ClickButton"(버튼 마우스 클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickButton()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickLabel

컨트롤 "ClickLabel"(라벨을 마우스 클릭) 내부 이벤트의 가상 핸들러

```
virtual bool  OnClickLabel()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CCheckGroup

CCheckGroup은 CheckGroup 복합 컨트롤(종속 컨트롤 포함)의 클래스입니다.

Description

CCheckGroup은 플래그를 표시하고 편집할 수 있는 컨트롤을 만들 수 있습니다.

Declaration

```
class CCheckGroup : public CWndClient
```

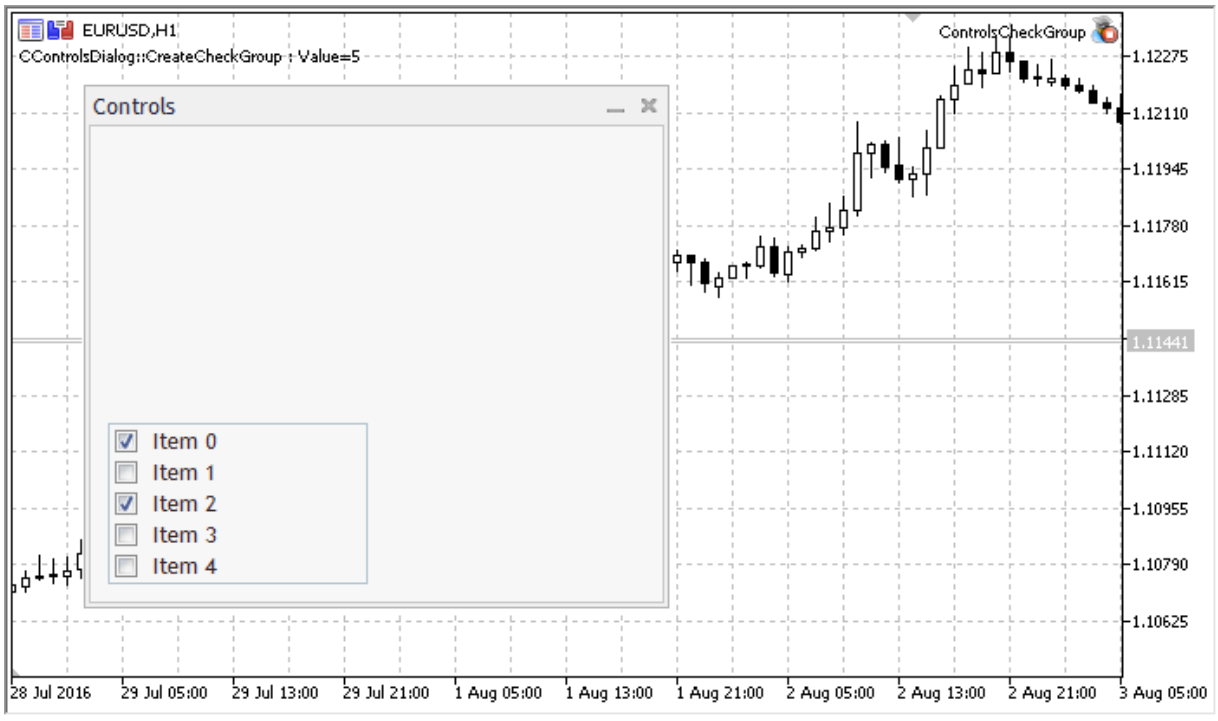
Title

```
#include <Controls\CheckGroup.mqh>
```

상속 계층

CObject
CWnd
CWndContainer
CWndClient
CCheckGroup

아래에 제시된 코드의 결과:



그룹별 클래스 메서드

© 2000-2025, MetaQuotes Ltd.

Create	
Create	컨트롤 생성
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트에 대한 이벤트 핸들러
Add	
AddItem	새 항목 추가
읽기전용 데이터	
Value	컨트롤과 관련된 값을 가져옵니다
종속 제어	
CreateButton	지정된 인덱스에 새 CCheckBox 항목을 만듭니다.
종속 컨트롤 이벤트 핸들러	
OnVScrollShow	VScroll 종속 컨트롤의 "Show" 내부 이벤트 핸들러(가상)
OnVScrollHide	VScroll 종속 컨트롤의 "Hide" 내부 이벤트 핸들러(가상)
OnScrollLineDown	VScroll 종속 컨트롤의 "ScrollLineUp" 내부 이벤트 핸들러(가상)
OnScrollLineUp	VScroll 종속 컨트롤의 "ScrollLineDown" 내부 이벤트 핸들러(가상)
OnChangeItem	VScroll 종속 컨트롤의 "ChangeItem" 내부 이벤트 핸들러(가상)
Redraw	
Redraw	그룹 리드로우(redraw)
RowState	그룹의 지정된 항목의 상태를 설정합니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

CWndClient 클래스에서 상속된 메서드

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

확인란 그룹 제어를 사용하여 패널을 만드는 예:

© 2000-2025, MetaQuotes Ltd.

```
//+------------------------------------------------------------------+
//|                                            ControlsCheckGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CCheckGroup"
#include <Controls\Dialog.mqh>
#include <Controls\CheckGroup.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckGroup       m_check_group;                  // CCheckGroup 객체

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
```

© 2000-2025, MetaQuotes Ltd.

```
   //--- 종속 컨트롤 생성
   bool              CreateCheckGroup(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnChangeCheckGroup(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_group,OnChangeCheckGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateCheckGroup())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "CheckGroup" 요소를 생성합니다                                   |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckGroup(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (RADIO_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
```

© 2000-2025, MetaQuotes Ltd.

```
   int y2=y1+CHECK_HEIGHT;
//--- 생성
   if(!m_check_group.Create(m_chart_id,m_name+"CheckGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_check_group))
      return(false);
//--- 문자열로 채웁니다
   for(int i=0;i<5;i++)
      if(!m_check_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_check_group.Check(0,1<<0);
   m_check_group.Check(2,1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(ChartID(),"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

새 CCheckGroup 컨트롤을 만듭니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

AddItem

컨트롤에 항목(행)을 추가합니다.

```
bool  AddItem(
   const string  item,     // 텍스트
   const long    value     // 값
   )
```

Parameters

item

[in] 추가된 컨트롤 레이블 텍스트.

value=0

[in] 추가된 컨트롤과 관련된 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Value

컨트롤과 관련된 값을 가져옵니다.

```
long  Value()
```

Return Value

컨트롤과 관련된 값.

Note

값은 CCheckGroup의 모든 항목의 상태에 따라 달라집니다.

© 2000-2025, MetaQuotes Ltd.

CreateButton

지정된 인덱스에 새 CCheckBox 클래스 인스턴스를 만듭니다.

```
bool  CreateButton(
   int  index      // 인덱스
   )
```

Parameters

index

[in] CCheckGroup의 새 항목 색인.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

종속 VScroll(수직 스크롤) 컨트롤 "Show" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnVScrollShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

종속 VScroll(수직 스크롤) 컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnVScrollHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

종속 VScroll(수직 스크롤)의 가상 핸들러는 "ScrollLineDown" 내부 이벤트를 제어합니다.

```
virtual bool  OnScrollLineDown()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

종속 VScroll(수직 스크롤) 컨트롤 "ScrollLineUp" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnScrollLineUp()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChangeItem

컨트롤 "ChangeItem"(항목 변경) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnChangeItem(
   const int  index      // 인덱스
   )
```

Parameters

index

[in] 변경된 아이템의 색인.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Redraw

컨트롤 다시 그리기.

```
bool  Redraw()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

RowState

지정한 항목의 상태를 설정합니다.

```
bool  RowState(
   const int   index,      // 항목 색인
   const bool  select      // 상태
   )
```

Parameters

index

[in] 변경할 항목 색인.

select

[in] 새 상태.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CRadioButton

CRadioButtonは RadioButton 복합 컨트롤의 한 종류입니다.

Description

CRadioButton 자체는 사용되지 않으며, CRadioGroup 항목을 만드는데 사용됩니다.

Declaration

```
class CRadioButton : public CWndContainer
```

Title

```
#include <Controls\RadioButton.mqh>
```

상속 계층

CObject
  CWnd
    CWndContainer
      CRadioButton

그룹별 클래스 메서드

Create	
Create	컨트롤 생성
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트에 대한 이벤트 핸들러
Properties	
Text	컨트롤과 연관된 문자 레이블을 가져오거나 설정합니다
Color	컨트롤과 연관된 문자 레이블 색상을 가져오거나 설정합니다
State	
State	상태를 가져오거나 설정합니다
종속 제어	
CreateButton	Creates button
CreateLabel	Creates label
종속 컨트롤 이벤트 핸들러	
OnClickButton	"ClickButton" 내부 이벤트 핸들러 (가상)
OnClickLabel	"ClickLabel" 내부 이벤트 핸들러 (가상)

© 2000-2025, MetaQuotes Ltd.

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

© 2000–2025, MetaQuotes Ltd.

Create

새 CRadioButton 컨트롤을 생성합니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Text (Get method)

컨트롤과 연결된 레이블의 텍스트를 가져옵니다.

```
string  Text()
```

Return Value

레이블 텍스트.

Text (Set method)

레이블의 텍스트를 설정합니다.

```
bool  Text(
   const string  value      // 텍스트
   )
```

Parameters

value

[in] 레이블의 새 텍스트.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Color (Get method)

컨트롤과 연결된 레이블의 색상을 가져옵니다.

```
color  Color()  const
```

Return Value

레이블 색상.

Color (Set method)

컨트롤과 연결된 레이블의 색상을 설정합니다.

```
bool  Color(
   const color  value      // 색상
   )
```

Parameters

value

[in] 새 레이블 색상.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

State (Get method)

버튼 상태를 가져옵니다.

```
bool  State()  const
```

Return Value

버튼 상태.

State (Set method)

버튼 상태를 설정합니다.

```
bool  State(
   const bool  flag      // 플래그
   )
```

Parameters

flag

[in] 새 버튼 상태.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateButton

버튼을 생성합니다.

```
virtual bool  CreateButton()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateLabel

레이블을 생성합니다.

```
virtual bool  CreateLabel()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickButton

컨트롤 "ClickButton"(버튼 클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickButton()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickLabel

"ClickLabel"(라벨 클릭) 컨트롤의 가상 핸들러 내부 이벤트.

```
virtual bool  OnClickLabel()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CRadioGroup

CRadioGroup은 RadioGroup 복합 제어(종속 제어 포함)의 한 클래스입니다.

Description

CRadioGroup을 사용하면 열거 가능한 유형 필드를 표시하고 편집할 수 있는 컨트롤을 만들 수 있습니다.

Declaration

```
class CRadioGroup : public CWndClient
```

Title

```
#include <Controls\RadioGroup.mqh>
```

상속 계층

CObject
CWnd
CWndContainer
CWndClient
CRadioGroup

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

© 2000-2025, MetaQuotes Ltd.

Create	
Create	컨트롤 생성
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Add	
AddItem	새 항목 추가
읽기전용 데이터	
Value	컨트롤과 관련된 값을 가져옵니다
종속 제어	
CreateButton	지정된 인덱스에 새 CRadioButton 항목을 작성합니다
종속 컨트롤 이벤트 핸들러	
OnVScrollShow	VScroll 종속 컨트롤의 "Show" 내부 이벤트 핸들러(가상)
OnVScrollHide	VScroll 종속 컨트롤의 "Hide" 내부 이벤트 핸들러(가상)
OnScrollLineDown	VScroll 종속 컨트롤의 "ScrollLineDown" 내부 이벤트 핸들러(가상)
OnScrollLineUp	VScroll 종속 컨트롤의 "ScrollLineUp" 내부 이벤트 핸들러(가상)
OnChangeItem	"ChangeItem" 내부 이벤트 처리기(가상)
Redraw	
Redraw	항목 그룹 다시 그리기
RowState	지정한 항목의 상태를 설정합니다
Select	현재 항목을 선택합니다

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

CWndClient 클래스에서 상속된 메서드

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

© 2000-2025, MetaQuotes Ltd.

라디오 단추 그룹이 있는 패널 만들기 예:

```
//+------------------------------------------------------------------+
//|                                          ControlsRadioGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CRadioGroup"
#include <Controls\Dialog.mqh>
#include <Controls\RadioGroup.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CRadioGroup       m_radio_group;                  // CRadioGroup 객체

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
```

© 2000-2025, MetaQuotes Ltd.

```
protected:
   //--- 종속 컨트롤 생성
   bool              CreateRadioGroup(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnChangeRadioGroup(void);
  };
//+------------------------------------------------------------------+
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_radio_group,OnChangeRadioGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateRadioGroup())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "RadioGroup" 요소를 생성                                         |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateRadioGroup(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (BUTTON_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y)+
          (EDIT_HEIGHT+CONTROLS_GAP_Y);
```

© 2000-2025, MetaQuotes Ltd.

```
   int x2=x1+GROUP_WIDTH;
   int y2=y1+RADIO_HEIGHT;
//--- 생성
   if(!m_radio_group.Create(m_chart_id,m_name+"RadioGroup",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_radio_group))
      return(false);
//--- 문자열로 채웁니다
   for(int i=0;i<3;i++)
      if(!m_radio_group.AddItem("Item "+IntegerToString(i),1<<i))
         return(false);
   m_radio_group.Value(1<<2);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeRadioGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
```

© 2000-2025, MetaQuotes Ltd.

```
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

새 CRadioGroup 컨트롤을 만듭니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

AddItem

컨트롤에 항목을 추가합니다.

```
bool  AddItem(
   const string  item,     // 텍스트
   const long    value=0   // 값
   )
```

Parameters

item

[in] 추가된 컨트롤 레이블 텍스트.

value=0

[in] 추가된 컨트롤과 관련된 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Value

컨트롤과 관련된 값을 가져옵니다.

```
long  Value()
```

Return Value

컨트롤과 관련된 값.

Note

값은 CRadioGroup 컨트롤의 모든 CRadioButton 항목의 상태에 따라 달라집니다.

© 2000-2025, MetaQuotes Ltd.

CreateButton

지정한 인덱스에 새 CRadioButton 클래스 인스턴스를 만듭니다.

```
bool  CreateButton(
   const int  index      // 인덱스
   )
```

Parameters

index

[in] CRadioGroup의 새 항목 인덱스.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollShow

종속 VScroll(수직 스크롤) 컨트롤 "Show" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnVScrollShow()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnVScrollHide

종속 VScroll(수직 스크롤) 컨트롤 "Hide" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnVScrollHide()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineDown

종속 VScroll(수직 스크롤)의 가상 핸들러는 "ScrollLineDown" 내부 이벤트를 제어합니다.

```
virtual bool  OnScrollLineDown()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnScrollLineUp

종속 VScroll(수직 스크롤) 컨트롤 "ScrollLineUp" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnScrollLineUp()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChangeItem

컨트롤 "ChangeItem"(항목 변경) 이벤트의 가상 핸들러.

```
virtual bool  OnChangeItem(
   const int  index      // 인덱스
   )
```

Parameters

index

[in] 변경된 아이템의 색인.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Redraw

컨트롤 다시 그리기.

```
bool  Redraw()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

RowState

항목의 상태를 변경합니다.

```
bool  RowState(
   const int   index,      // 인덱스
   const bool  select      // 상태
   )
```

Parameters

index

[in] 변경할 항목 색인.

select

[in] 새 상태.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Select

현재 항목을 선택합니다.

```
void  Select(
   const int  index      // 인덱스
   )
```

Parameters

index

[in] 선택할 아이템 색인.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

CSpinEdit

CSpinEdit는 SpinEdit 복합 컨트롤의 한 종류입니다.

Description

CSpinEdit 클래스는 지정된 단계로 지정된 제한 내에서 정수 형식의 값을 편집할 수 있는 컨트롤을 만들기 위한 것입니다.

Declaration

```
class CSpinEdit : public CWndContainer
```

Title

```
#include <Controls\SpinEdit.mqh>
```

상속 계층

COBject
CWnd
CWndContainer
CSpinEdit

아래에 제시된 코드의 결과:

그룹별 클래스 메서드

Create	
Create	컨트롤 생성

© 2000-2025, MetaQuotes Ltd.

Create	
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Properties	
MinValue	허용되는 최소값을 가져오거나 설정합니다
MaxValue	허용하는 최대값을 가져오거나 설정합니다
State	
Value	현재 값을 가져오거나 설정합니다
종속 제어	
CreateEdit	종속 컨트롤을 만듭니다(편집)
CreateInc	종속 컨트롤(증분 버튼)을 작성합니다
CreateDec	종속 컨트롤(감쇠 버튼)을 작성합니다
종속 컨트롤 이벤트 핸들러	
OnClickInc	"ClickInc" 내부 이벤트 핸들러(가상)
OnClickDec	"ClickDec" 내부 이벤트 핸들러(가상)
내부 이벤트 핸들러	
OnChangeValue	"ChangeValue" 내부 이벤트 핸들러(가상)

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

스핀 편집 컨트롤을 사용한 패널 작성 예제:

```
//+------------------------------------------------------------------+
//|                                            ControlsSpinEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
```

© 2000-2025, MetaQuotes Ltd.

```
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "제어판 및 대화 상자. 데모 클래스 CSpinEdit"
#include <Controls\Dialog.mqh>
#include <Controls\SpinEdit.mqh>
//+------------------------------------------------------------------+
//| 정의                                                             |
//+------------------------------------------------------------------+
//--- 틈
#define INDENT_LEFT                         (11)      // 왼쪽에서 들여쓰기(경계 너비 허용)
#define INDENT_TOP                          (11)      // 맨 위에서 들여쓰기(경계 너비 허용
#define INDENT_RIGHT                        (11)      // 오른쪽에서 들여쓰기(경계 너비 허용
#define INDENT_BOTTOM                       (11)      // 아래쪽에서 들여쓰기(경계 너비 허용
#define CONTROLS_GAP_X                      (5)       // X 좌표에 의한 차이
#define CONTROLS_GAP_Y                      (5)       // Y 좌표에 의한 차이
//--- 버튼용
#define BUTTON_WIDTH                        (100)     // X 좌표에 의한 크기
#define BUTTON_HEIGHT                       (20)      // Y 좌표에 의한 크기
//--- 표시 영역용
#define EDIT_HEIGHT                         (20)      // Y 좌표에 의한 크기
//--- 그룹 컨트롤용
#define GROUP_WIDTH                         (150)     // X 좌표에 의한 크기
#define LIST_HEIGHT                         (179)     // Y 좌표에 의한 크기
#define RADIO_HEIGHT                        (56)      // Y 좌표에 의한 크기
#define CHECK_HEIGHT                        (93)      // Y 좌표에 의한 크기
//+------------------------------------------------------------------+
//| Class CControlsDialog                                            |
//| Usage: 제어 응용 프로그램의 주 대화 상자                        |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CSpinEdit         m_spin_edit;                    // CSpinEdit object

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- 생성
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- 차트 이벤트 핸들러
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- 종속 컨트롤 생성
   bool              CreateSpinEdit(void);
   //--- 종속 제어 이벤트 핸들러
   void              OnChangeSpinEdit(void);
  };
//+------------------------------------------------------------------+
```

© 2000-2025, MetaQuotes Ltd.

```
//| 이벤트 처리                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
   ON_EVENT(ON_CHANGE,m_spin_edit,OnChangeSpinEdit)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| 생성자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 소멸자                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| 생성                                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::Create(const long chart,const string name,const int subwin,const
  {
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- 종속 컨트롤 생성
   if(!CreateSpinEdit())
      return(false);
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| "SpinEdit" 요소 생성                                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateSpinEdit(void)
  {
//--- 좌표
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y)+(BUTTON_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+GROUP_WIDTH;
   int y2=y1+EDIT_HEIGHT;
//--- 생성
   if(!m_spin_edit.Create(m_chart_id,m_name+"SpinEdit",m_subwin,x1,y1,x2,y2))
      return(false);
   if(!Add(m_spin_edit))
      return(false);
   m_spin_edit.MinValue(10);
   m_spin_edit.MaxValue(100);
   m_spin_edit.Value(50);
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
```

© 2000-2025, MetaQuotes Ltd.

```
//--- 성공
   return(true);
  }
//+------------------------------------------------------------------+
//| 이벤트 핸들러                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeSpinEdit(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
  }
//+------------------------------------------------------------------+
//| 글로벌 변수                                                      |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert 초기화 함수                                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 응용 프로그램 대화 상자 만들기
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- 어플리케이션 실행
   ExtDialog.Run();
//--- 성공
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert 초기화 해제 함수                                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- 댓글 삭제
   Comment("");
//--- 대화 상자 소멸
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert 차트 이벤트 함수                                          |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // 이벤트 ID
                  const long& lparam,   // long 타입의 이벤트 매개 변수
                  const double& dparam, // double 타입의 이벤트 매개 변수
                  const string& sparam) // string 타입의 이벤트 매개 변수
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

© 2000-2025, MetaQuotes Ltd.

Create

새 CSpinEdit 컨트롤을 만듭니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

MinValue (Get method)

컨트롤의"MinValue" 특성 값(최소값)을 가져옵니다.

```
int  MinValue()  const
```

Return Value

"MinValue" 속성 값.

MinValue (Set method)

컨트롤의 "최소값" 속성 값(최소값)을 설정합니다.

```
void  MinValue(
   const int  value      // 값
   )
```

Parameters

value

[in] "MinValue" 속성의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

MaxValue (Get method)

컨트롤의 "MaxValue" 속성 값(최대값)을 가져옵니다.

```
int  MaxValue()  const
```

Return Value

"MaxValue" 속성의 값.

MaxValue (Set method)

컨트롤의 "MaxValue" 속성 값(최대값)을 설정합니다.

```
void  MaxValue(
   const int  value      // 값
   )
```

Parameters

value

[in] "MaxValue" 속성의 새 값.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Value (Get method)

컨트롤의 "값" 특성(현재 값)을 가져옵니다.

```
int  Value()  const
```

Return Value

"Value" 속성.

Value (Set method)

컨트롤의 "Value" 속성(현재 값)을 설정합니다.

```
void  Value(
   const int  value      // 값
   )
```

Parameters

value

[in] 새 "Value" 속성.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

CreateEdit

종속 컨트롤(CEDIT)을 생성합니다.

```
virtual bool  CreateEdit()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateInc

종속 컨트롤(증분 단추)을 생성합니다.

```
virtual bool  CreateInc()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateDec

종속 컨트롤(변환 버튼)을 만듭니다.

```
virtual bool  CreateDec()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickInc

컨트롤 "ClickInc"(증분 버튼을 마우스로 클릭) 내부 이벤트의 가상 처리기.

```
virtual bool  OnClickInc()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickDec

컨트롤 "ClickDec"(감소 단추를 마우스로 클릭) 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickDec()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnChangeValue

"ChangeValue" 컨트롤의 가상 핸들러(현재 값 변경) 내부 이벤트.

```
virtual bool  OnChangeValue()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CDialog

CDialog는 Dialog 복합 컨트롤의 클래스입니다.

Description

CDialog 클래스는 컨트롤을 그룹의 다른 함수와 결합하기 위한 것입니다.

Declaration

```
class CDialog : public CWndContainer
```

Title

```
#include <Controls\Dialog.mqh>
```

상속 계층

CObject
  CWnd
    CWndContainer
      CDialog

직계 종속

CAppDialog

그룹별 클래스 메서드

Create	
Create	컨트롤 생성
차트 이벤트 핸들러	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Properties	
Caption	"Caption" 속성 값을 가져오거나 설정합니다.
Add	
Add	클라이언트 영역에 컨트롤을 추가합니다
종속 제어	
CreateWhiteBorder	종속 컨트롤을 만듭니다(흰색 테두리)
CreateBackground	종속 컨트롤을 만듭니다(배경)
CreateCaption	종속 컨트롤을 만듭니다(캡션)
CreateButtonClose	종속 컨트롤을 만듭니다(닫기 버튼)
CreateClientArea	종속 컨트롤을 만듭니다(클라이언트 영역)

© 2000-2025, MetaQuotes Ltd.

Create	
종속 컨트롤 이벤트 핸들러	
OnClickCaption	"ClickCaption" 내부 이벤트 핸들러
OnClickButtonClose	"ClickButtonClose" 내부 이벤트 핸들러
클라이언트 영역 속성에 대한 액세스	
ClientAreaVisible	클라이언트 영역의 표시 여부를 나타내는 값을 설정합니다
ClientAreaLeft	컨트롤 클라이언트 영역의 왼쪽 상단 모서리의 X 좌표를 가져옵니다
ClientAreaTop	컨트롤 클라이언트 영역의 왼쪽 상단 모서리의 Y 좌표를 가져옵니다
ClientAreaRight	컨트롤 클라이언트 영역의 오른쪽 하단 모서리의 X 좌표를 가져옵니다
ClientAreaBottom	컨트롤 클라이언트 영역의 오른쪽 하단 모서리의 Y 좌표를 가져옵니다
ClientAreaWidth	클라이언트 영역 폭을 가져옵니다
ClientAreaHeight	클라이언트 영역 높이를 가져옵니다
이벤트 핸들러 드래그	
OnDialogDragStart	"DialogDragStart" 이벤트 핸들러(가상)
OnDialogDragProcess	"DialogDragProcess" 이벤트 핸들러(가상)
OnDialogDragEnd	"DialogDragEnd" 이벤트 핸들러(가상)

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

© 2000-2025, MetaQuotes Ltd.

Create

새 CDialog 컨트롤을 생성합니다.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Caption (Get method)

CDialog 컨트롤의 "Caption" 속성을 가져옵니다.

```
string  MinValue()  const
```

Return Value

"Caption" 속성.

Caption (Set method)

CDialog 컨트롤의 "Caption" 속성을 설정합니다.

```
bool  Caption(
   const string  text      // 텍스트
   )
```

Parameters

text

[in] "Caption" 속성의 새 값.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Add

포인터로 클라이언트 영역에 컨트롤을 추가합니다.

```
bool  Add(
   CWnd   *control,        // 포인터
   )
```

Parameters

control

[in] 컨트롤로의 포인터.

Return Value

성공시 true, 아니면 false.

Add

참조로 클라이언트 영역에 컨트롤을 추가합니다.

```
bool  Add(
   CWnd   &control,        // 레퍼런스
   )
```

Parameters

control

[in] 컨트롤로의 레퍼런스.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateWhiteBorder

종속 컨트롤을 생성합니다(흰색 테두리).

```
virtual bool  CreateWhiteBorder()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateBackground

종속 제어를 생성합니다(배경).

```
virtual bool  CreateBackground()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateCaption

종속 컨트롤을 생성합니다(캡션).

```
virtual bool  CreateCaption()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateButtonClose

종속 컨트롤을 만듭니다(닫기 버튼)

```
virtual bool  CreateButtonClose()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateClientArea

종속 컨트롤을 생성합니다(클라이언트 영역).

```
virtual bool  CreateClientArea()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickCaption

컨트롤 "ClickCaption" 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickCaption()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose

"ClickButtonClose" 컨트롤 내부 이벤트의 가상 핸들러.

```
virtual bool  OnClickButtonClose()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ClientAreaVisible

클라이언트 영역의 표시 여부를 나타내는 플래그를 설정합니다.

```
bool  ClientAreaVisible(
   const bool  visible      // 가시성 플래그
   )
```

Parameters

visible

[in] 가시성 플래그.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ClientAreaLeft

컨트롤 클라이언트 영역의 왼쪽 상단 모서리에 대한 X 좌표를 가져옵니다.

```
int  ClientAreaLeft()
```

Return Value

컨트롤 클라이언트 영역의 왼쪽 상단 모서리의 X 좌표.

© 2000-2025, MetaQuotes Ltd.

ClientAreaTop

컨트롤 클라이언트 영역의 왼쪽 상단 모서리의 Y 좌표를 가져옵니다.

```
int  ClientAreaTop()
```

Return Value

컨트롤 클라이언트 영역의 왼쪽 상단 모서리의 Y 좌표.

© 2000-2025, MetaQuotes Ltd.

ClientAreaRight

컨트롤 클라이언트 영역의 오른쪽 하단 모서리의 X 좌표를 가져옵니다.

```
int  ClientAreaTop()
```

Return Value

컨트롤 클라이언트 영역의 오른쪽 하단 모서리의 X 좌표.

© 2000-2025, MetaQuotes Ltd.

ClientAreaBottom

컨트롤 클라이언트 영역의 오른쪽 하단 모서리의 Y 좌표를 가져옵니다.

```
int  ClientAreaBottom()
```

Return Value

제어 클라이언트 영역의 오른쪽 하단 모서리의 Y 좌표.

© 2000-2025, MetaQuotes Ltd.

ClientAreaWidth

제어 클라이언트 영역의 너비를 가져옵니다.

```
int  ClientAreaWidth()
```

Return Value

클라이언트 영역의 너비.

© 2000-2025, MetaQuotes Ltd.

ClientAreaHeight

컨트롤 클라이언트 영역의 높이를 가져옵니다.

```
int  ClientAreaHeight()
```

Return Value

컨트롤 클라이언트 영역의 높이.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragStart

컨트롤 "DialogDragStart" 이벤트의 가상 핸들러.

```
virtual bool  OnDialogDragStart()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

컨트롤 끌기가 시작될 때 "DialogDragStart" 이벤트가 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragProcess

컨트롤 "DialogDragProcess" 이벤트의 가상 핸들러.

```
virtual bool  OnDialogDragProcess()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

컨트롤을 드래그 하면 "DialogDragProcess" 이벤트가 발생합니다.

© 2000-2025, MetaQuotes Ltd.

OnDialogDragEnd

컨트롤 "DialogDragEnd" 이벤트의 가상 핸들러.

```
virtual bool  OnDialogDragEnd()
```

Return Value

이벤트가 처리되면 true, 아니면 false.

Note

"DialogDragEnd" 이벤트는 컨트롤 드래그의 끝에서 발생합니다.

© 2000-2025, MetaQuotes Ltd.

CAppDialog

CAppDialog는 종속 컨트롤이 있는 응용 프로그램 대화 상자 복합 컨트롤의 클래스.

Description

CAppDialog 클래스는 MQL5 프로그램 내의 그룹에서 서로 다른 함수를 가진 컨트롤을 결합하기 위한 것입니다.

Declaration

```
class CAppDialog : public CDialog
```

Title

```
#include <Controls\Dialog.mqh>
```

상속 계층

CObject
  CWnd
    CWndContainer
      CDialog
        CAppDialog

그룹별 클래스 메서드

생성 및 제거	
Create	컨트롤 생성
Destroy	컨트롤 제거
이벤츠 터리	
OnEvent	모든 차트 이벤트의 이벤트 핸들러
Run	
Run	컨트롤 실행
차트 이벤트 처리	
ChartEvent	모든 차트 이벤트의 이벤트 핸들러
Settings	
Minimized	컨트롤의 최소화 여부를 나타내는 값을 설정합니다
Save/Load	
IniFileSave	컨트롤 상태를 파일에 저장합니다
IniFileLoad	파일에서 컨트롤 상태를 로드합니다
IniFileName	제어 상태를 로드/저장하기 위한 파일 이름을 설정합니다

© 2000-2025, MetaQuotes Ltd.

생성 및 제거	
IniFileExt	제어 상태를 로드/저장하기 위한 파일 확장자를 설정합니다
Initialization	
CreateCommon	공통 초기화 방법
CreateExpert	Expert Advisor 작업을 위한 초기화 방법
CreateIndicator	지표 작업을 위한 초기화 방법
종속 제어	
CreateButtonMinMax	종속 제어 만들기(버튼 최소화/최대화)
종속 컨트롤 이벤트 핸들러	
OnClickButtonClose	"ClickButtonClose" 내부 이벤트 핸들러(가상)
OnClickButtonMinMax	"ClickButtonMinMax" 내부 이벤트 핸들러(가상)
외부 이벤트	
OnAnotherApplicationClose	외부 이벤트의 이벤트 핸들러(가상)
Methods	
Rebound	CRect 클래스 좌표를 사용하여 컨트롤의 새 좌표를 설정합니다
Minimize	컨트롤을 최소화된 상태로 표시합니다
Maximize	컨트롤을 최대화(복원) 상태로 표시합니다.
CreateInstanceId	컨트롤 개체의 이름에 대한 고유한 ID를 만듭니다
ProgramName	컨트롤이 사용되는 MQL5 프로그램의 이름을 가져옵니다
SubwinOff	컨트롤 하위 창의 Y 오프셋 가져오기

클래스 CObject에서 상속된 메서드

Prev, Prev, Next, Next, Type, Compare

클래스 CWnd에서 상속된 메서드

Name, ControlsTotal, Control, Rect, Left, Left, Top, Top, Right, Right, Bottom, Bottom, Width, Width, Height, Height, Size, Size, Size, Contains, Contains, Alignment, Align, Id, IsEnabled, IsVisible, Visible, IsActive, Activate, Deactivate, StateFlags, StateFlags, StateFlagsSet, StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

클래스 CWndContainer에서 상속된 메서드

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

클래스 CDialog에서 상속된 메서드

Caption, Caption, Add, Add

© 2000-2025, MetaQuotes Ltd.

© 2000-2025, MetaQuotes Ltd.

Create

새 CAppDialog 컨트롤 생성.

```
virtual bool  Create(
   const long     chart,      // 차트 ID
   const string   name,       // 이름
   const int      subwin,     // 차트 하위창
   const int      x1,         // 좌표
   const int      y1,         // 좌표
   const int      x2,         // 좌표
   const int      y2          // 좌표
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Destroy

CAppDialog 컨트롤 초기화 해제 메서드.

```
virtual void  Destroy(
   const int  reason=REASON_PROGRAM    // 리즌 코드
)
```

Parameters

reason

[in] 초기화 해제 리즌 코드. REASON_PROGRAM이 기본적으로 설정됩니다.

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

OnEvent

차트 이벤트 핸들러.

```
virtual bool  OnEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Run

컨트롤 실행.

```
bool  Run()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

ChartEvent

컨트롤 이벤트의 가상 핸들러.

```
virtual bool  ChartEvent(
   const int      id,          // ID
   const long&    lparam,      // 매개 변수
   const double&  dparam,      // 매개 변수
   const string&  sparam       // 매개 변수
   )
```

Parameters

id

[in] Event ID.

lparam

[in] 참조로 전달된 long 타입의 이벤트 매개 변수.

dparam

[in] 참조로 전달된 double 타입의 이벤트 매개 변수.

sparam

[in] 참조로 전달된 string 유형의 이벤트 매개 변수.

Return Value

이벤트가 처리되면 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Minimized

컨트롤의 "최소화"(창 상태) 속성 값을 설정합니다.

```
bool  Minimized(
   const bool  flag      // 상태
   )
```

Parameters

flag

[in] 새 상태.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

IniFileSave

컨트롤 상태를 파일로 저장합니다.

```
void  IniFileSave()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

IniFileLoad

파일에서 제어 상태를 로드합니다.

```
void  IniFileLoad()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

IniFileName

제어 상태를 로드/저장하기 위한 파일 이름을 설정합니다.

```
virtual string  IniFileName()  const
```

Return Value

컨트롤 상태를 로드/저장하기 위한 파일 이름.

Note

파일 이름에는 MQL5 프로그램이 실행되는 Expert Advisor/지표 및 작업 심볼의 이름이 포함됩니다.

© 2000-2025, MetaQuotes Ltd.

IniFileExt

컨트롤 상태를 로드/저장하기 위한 파일 확장자를 설정합니다.

```
virtual string  IniFileExt()  const
```

Return Value

제어 상태를 로드/저장하는 데 사용되는 파일 확장명.

© 2000-2025, MetaQuotes Ltd.

CreateCommon

공통 초기화 방법.

```
bool  CreateCommon(
   const long    chart,      // 차트 ID
   const string  name,       // 이름
   const int     subwin,     // 차트 하위창
   )
```

Parameters

chart

[in] 컨트롤이 생성되는 차트의 ID.

name

[in] 컨트롤의 고유 이름.

subwin

[in] 컨트롤이 생성되는 차트의 하위창.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateExpert

Expert Advisors에서 작업하기 위한 초기화 방법.

```
bool  CreateExpert(
   const int      x1,          // 좌표
   const int      y1,          // 좌표
   const int      x2,          // 좌표
   const int      y2           // 좌표
   )
```

Parameters

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateIndicator

지표 작업을 위한 초기화 방법.

```
bool  CreateIndicator(
   const int      x1,          // 좌표
   const int      y1,          // 좌표
   const int      x2,          // 좌표
   const int      y2           // 좌표
   )
```

Parameters

x1

[in] 왼쪽 위 모서리의 X 좌표.

y1

[in] 왼쪽 위 모서리의 Y 좌표.

x2

[in] 오른쪽 아래 모서리의 X 좌표.

y2

[in] 오른쪽 아래 모서리의 Y 좌표.

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateButtonMinMax

종속 컨트롤을 만듭니다(버튼 최소화/최대화).

```
virtual void  CreateButtonMinMax()
```

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonClose

컨트롤 "ClickButtonClose"(닫기 버튼에 마우스 클릭) 내부 이벤트의 가상 핸들러.

```
virtual void  OnClickButtonClose()
```

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

OnClickButtonMinMax

컨트롤 "ClickButtonMinMax"(최소화/최대화 버튼 마우스 클릭) 내부 이벤트의 가상 핸들러.

```
virtual void  OnClickButtonClose()
```

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

OnAnotherApplicationClose

컨트롤 외부 이벤트의 가상 핸들러.

```
virtual void  OnAnotherApplicationClose()
```

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

Rebound

CRect 클래스 좌표를 사용하여 컨트롤의 새 좌표를 설정합니다.

```
bool  Rebound(
   const & CRect  rect      // CRect 클래스
   )
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Minimize

최소화된 상태의 제어 창을 표시합니다.

```
virtual void  Minimize()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

Maximize

제어 창을 최대화(복원) 상태로 표시합니다.

```
virtual void  Maximize()
```

Return Value

성공시 true, 아니면 false.

© 2000-2025, MetaQuotes Ltd.

CreateInstanceId

컨트롤 개체 이름에 대한 고유한 접두사를 만듭니다.

```
string  CreateInstanceId()
```

Return Value

개체 이름의 접두사.

© 2000-2025, MetaQuotes Ltd.

ProgramName

컨트롤이 사용되는 MQL5 프로그램의 이름을 가져옵니다.

```
string  ProgramName()
```

Return Value

MQL5 프로그램의 이름.

© 2000-2025, MetaQuotes Ltd.

SubwinOff

컨트롤 하위 창의 Y 오프셋을 가져옵니다.

```
void  SubwinOff()
```

Return Value

None.

© 2000-2025, MetaQuotes Ltd.

MQL4에서 MQL5로 이전

MQL5는 MQL4 프로그래밍 언어의 이전 버전인 MQL4 프로그래밍 언어의 발전으로 수많은 지표, 스크립트 및 Expert Advisor가 작성되었습니다. 새로운 프로그래밍 언어가 이전 세대의 언어와 최대한 호환된다는 사실에도 불구하고, 이 언어들 사이에는 여전히 약간의 차이가 있습니다. 그리고 프로그램을 전송할 때 이러한 차이점에 유의해야 합니다.

이 섹션에는 MQL4를 알고 있는 프로그래머를 위해 새로운 MQL5 언어에 대한 코드 적용을 용이하게 하기 위한 정보가 수록되어 있습니다.

먼저 다음 사항에 유의해야 합니다.

- 새 언어에 start(), init() 및 deinit() 함수가 없습니다.
- 지표 버퍼 수에 제한이 없습니다.
- DLL은 Expert Advisor(또는 다른 mql5 프로그램)를 로드한 후 즉시 로드됩니다.
- 논리적 조건의 점검이 단축됩니다.
- 배열의 한계가 초과되면 현재 퍼포먼스가 종료됩니다(심각한 오류의 출력으로).
- C++와 같은 연산자의 우선 순위.
- 언어는 암묵적 유형의 타입 캐스트를 제공합니다(심지어 문자열부터 숫자까지).
- 로컬 변수는 문자열을 제외하고 자동으로 초기화되지 않습니다.
- 공통 로컬 배열은 자동으로 삭제됩니다.

특수 함수 init, start, deinit

MQL4 언어는 지표, 스크립트 또는 Expert Advisor에서 사용할 수 있는 세 가지 미리 정의된 함수만 포함하고 있습니다(*.mqh 및 라이브러리 파일 포함 고려하지 않음). MQL5에는 그러한 함수는 없지만, 그 유사점은 있습니다. 표에는 함수의 근사치가 나와 있습니다.

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

함수 OnInit 및 OnDeinit는 MQL4에서 init 및 deinit과 동일한 역할을 수행합니다. 이 함수는 mql5 프로그램의 초기화 및 초기화 중에 수행되어야 하는 코드를 찾기 위해 설계되었습니다. 이러한 함수의 이름을 적절하게 바꾸거나 그대로 둘 수 있지만 해당 위치에 해당 함수의 호출을 추가할 수 있습니다.

예:

```
void OnInit()
  {
//--- 초기화 시 함수 호출
   init();
  }
void OnDeinit(const int reason)
  {
//--- 초기화 취소 시 함수 호출
```

© 2000-2025, MetaQuotes Ltd.

```
   deinit();
//---
  }
```

시작 함수는 스크립트에서만 OnStart로 대체됩니다. Expert Advisor 및 지표에서 OnTick 및 OnCalculate로 이름을 각각 변경해야 합니다. mql5 프로그램 작동 중에 실행할 코드는 다음 세 가지 함수에 있어야 합니다.

mql5-program	main function
스크립트	OnStart
지표	OnCalculate
Expert Advisor	OnTick

지표 또는 스크립트 코드가 주 함수을 포함하지 않거나 함수명이 필요한 기능과 다를 경우 이 함수의 호출이 수행되지 않습니다. 즉, 스크립트의 소스 코드에 OnStart가 포함되어 있지 않으면 해당 코드가 Expert Advisor로 컴파일됩니다.

지표 코드에 OnCalculate 함수가 없는 경우 이러한 지표를 컴파일할 수 없습니다.

미리 정의된 변수

MQL5에는 Ask, Bid, Bars와 같은 사전 정의된 변수가 없습니다. 변수 포인트 및 숫자 스펠링은 약간의 차이가 있습니다.

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

시계열에 액세스

MQL5에는 Open[], High[], Low[], Close[], Volume[] 및 Time[]과 같은 사전 정의된 시계열은 없습니다. 이제 시계열의 필요한 깊이를 설정하여 시계열에 액세스할 수 있는 함수를 사용할 수 있습니다.

Expert Advisor

MQL5의 Expert Advisors는 MQL4에서와 같이 새로운 틱 수신 – OnTick의 이벤트를 처리하는 함수의 의무적인 존재는 요구하지 않습니다(새 눈금 수신 시 MQL4의 시작 함수가 실행됨). MQL5에서 Expert Advisor는 다음과 같은 여러 유형의 이벤트에 대해 미리 정의된 핸들러 함수를 포함할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

- OnTick – 새 틱의 수신;
- OnTimer – 타이머 이벤트;
- OnTrade – 트레이드 이벤트;
- OnChartEvent – 키보드와 마우스의 입력 이벤트, 그래픽 개체 이동 이벤트, LabelEdit 개체의 입력 필드에 있는 텍스트 편집 완료 이벤트;
- OnBookEvent – 마켓 상태 변경의 깊이에 대한 이벤트.

커스텀 지표

MQL4에서는 지표 버퍼 수가 제한되고 8을 초과할 수 없습니다. MQL5에서는 그러한 제한이 없지만, 각 지표 버퍼는 터미널에서 그것의 위치를 위해 메모리의 특정 부분을 할당해야 하기 때문에 새로운 가능성을 남용하지 않아야 한다는 것을 기억해야 합니다.

MQL4는 6가지 유형의 커스텀 지표 플로팅만 제공했지만 MQL5는 이제 18가지 그리기 스타일을 제공합니다. 도면 형식의 이름은 바뀌지 않았지만 지표의 그래픽 표현 이념은 크게 바뀌었습니다.

지표 버퍼의 색인화 방향도 다릅니다. 기본적으로 MQL5에서 모든 지표 버퍼는 공통 배열의 동작을 가지고 있습니다. 즉, 0 인덱스 요소는 내역에서 가장 오래된 요소이며 인덱스가 증가함에 따라 가장 오래된 데이터에서 가장 최근의 것으로 이동합니다.

MQL4에서 보존된 커스텀 지표로 작동하는 유일한 기능은 SetIndexBuffer입니다. 그러나 호출이 변경되었습니다. 이제 지표 버퍼에 연결된 배열에저장할 데이터 유형을 지정해야 합니다.

커스텀 지표의 속성도 변경 및 확장되었습니다. 시계열 접근을 위한 새로운 기능이 추가되었으므로 총 계산 알고리즘을 다시 고려해야 합니다.

그래픽 오브젝트

MQL5의 그래픽 개체 수가 크게 증가했습니다. 또한 그래픽 객체는 모든 시간대의 차트에서 1초의 정확도로 시간 내에 배치될 수 있습니다. 이제 객체 앵커 포인트는 현재 가격 차트에서 막대 오프닝 시간으로 반올림되지 않습니다.

이제 화살표, 텍스트 및 레이블 개체의 경우 바인딩 방법을 지정하고, 레이블, 단추, 차트, 비트맵 레이블 및 편집 개체의 경우 개체가 연결된 차트 모서리를 설정할 수 있습니다.

© 2000-2025, MetaQuotes Ltd.

MQL5 기능 목록

모든 MQL5 기능을 알파벳 순으로.

기능	액션	섹션
AccountInfoDouble	해당 계정 속성의 double 유형 값을 반환	계정 정보
AccountInfoInteger	해당 계정 속성의 정수 유형 (bool, int 또는 long) 값을 반환	계정 정보
AccountInfoString	해당 계정 속성의 string 유형 값을 반환	계정 정보
acos	라디안 단위 x의 아크 코사인을 반환	산술 함수
경고	별도의 창에 메시지를 표시	공통 함수
ArrayBsearch	오름차순으로 정렬된 다차원 수열에서 지정된 값을 검색	배열 함수
ArrayCompare	다음 두 배열을 비교한 결과를 반환:단순 유형 또는 맞춤형 구조 복합 개체	배열 함수
ArrayCopy	한개 배열을 다른 배열로 복사	배열 함수
ArrayFill	지정된 값으로 배열을 채우기	배열 함수
ArrayFree	모든 동적 배열의 버퍼를 해제하고 0 차원의 크기를 0으로 설정	배열 함수
ArrayGetAsSeries	배열 인덱싱 방향을 확인	배열 함수
ArrayInitialize	수열의 모든 요소를 단일 값으로 설정	배열 함수
ArrayIsDynamic	배열이 동적인지 아닌지 확인	배열 함수
ArrayIsSeries	배열이 시계열인지 아닌지 확인	배열 함수
ArrayMaximum	다차원 수열의 첫 번째 차원에서 가장 큰 요소를 검색	배열 함수
ArrayMinimum	다차원 수열의 첫 번째 차원에서 가장 낮은 요소를 검색	배열 함수
ArrayRange	배열의 지정된 차원에 있는 요소의 수를 반환	배열 함수
ArrayResize	베열의 첫 번째 차원에서 새로 크기를 설정	배열 함수
ArraySetAsSeries	배열 인텍싱의 방향을 설정	배열 함수
ArraySize	배열에서 요소의 수를 반환	배열 함수

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
ArraySort	첫 번째 차원을 기준으로 수열 정렬	배열 함수
ArrayPrint	단순 유형 또는 단순 구조의 배열을 저널에 인쇄	배열 함수
ArrayInsert	소스 배열에서 지정된 수의 요소를 지정된 인덱스에서 시작하여 수신하는 요소에 삽입	배열 함수
ArrayRemove	지정된 인덱스로 시작하는 배열에서 지정된 수의 요소를 제거	배열 함수
ArrayReverse	지정된 인덱스로 시작하는 배열에서 지정된 수의 요소를 반전	배열 함수
ArraySwap	동일 유형의 두 동적 배열 내용을 스왑	배열 함수
asin	라디안 단위로 x의 아크 사인을 반환	산술 함수
atan	라디안 단위의 x의 아크 탄젠트를 반환	산술 함수
막대	지정된 심볼 및 기간에 대한 내역의 막대 수를 반환	시계열 및 지표 액세스
BarsCalculated	오류 발생 시 지표 버퍼 또는 -1이 계산된 데이터 수를 반환 (데이터가 아직 계산되지 않음)	시계열 및 지표 액세스
CalendarCountryById	ID로 국가 설명을 가져오기	이코노믹 캘린더
CalendarEventById	ID로 이벤트 설명을 가져오기	이코노믹 캘린더
CalendarValueById	ID를 기준으로 이벤트 값 설명을 가져오기	이코노믹 캘린더
CalendarCountries	캘린더에서 사용할 수 있는 국가명 배열을 가져오기	이코노믹 캘린더
CalendarEventByCountry	캘린더에서 사용할 수 있는 모든 이벤트에 대한 설명 배열을 지정된 국가 코드로 가져오기	이코노믹 캘린더
CalendarEventByCurrency	캘린더에서 사용할 수 있는 모든 이벤트에 대한 설명 배열을 지정된 통화로 가져오기	이코노믹 캘린더
CalendarValueHistoryByEvent	지정된 시간 범위의 모든 이벤트에 대한 값 배열을 이벤트 ID를 기준으로 가져오기	이코노믹 캘린더

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
CalendarValueHistory	지정된 시간 범위의 모든 이벤트에 대한 값의 배열을 정렬할 수 있는 국가 및 통화별로 가져오기	이코노믹 캘린더
CalendarValueLastByEvent	change_id를 지정한 캘린더 데이터베이스 상태 이후 ID 별로 이벤트 값 배열을 가져오기	이코노믹 캘린더
CalendarValueLast	change_id를 지정한 캘린더 데이터베이스 상태 이후 국가 및 통화별로 정렬할 수 있는 모든 이벤트의 값 배열을 가져오기	이코노믹 캘린더
ceil	위에서 가장 가까운 정수 숫자 값을 반환	산술 함수
CharArrayToString	심볼 코드 (ansi)를 하나의 심볼 배열로 변환	변환 함수
ChartApplyTemplate	지정된 파일의 특정 템플릿을 차트에 적용	차트 작업
ChartClose	지정된 차트를 닫기	차트 작업
ChartFirst	클라이언트 터미널의 첫 번째 차트의 ID를 받환	차트 작업
ChartGetDouble	지정된 차트의 double 값 속성을 반환	차트 작업
ChartGetInteger	지정된 차트의 정수 값 속성을 반환	차트 작업
ChartGetString	지정한 차트의 string 값 속성을 반환	차트 작업
ChartID	현재 차트의 ID를 반화	차트 작업
ChartIndicatorAdd	지정된 핸들이 있는 지표를 지정된 차트 창에 추가	차트 작업
ChartIndicatorDelete	지정된 차트 창에서 지정된 이름의 지표를 제거	차트 작업
ChartIndicatorGet	지정된 차트 창에 지정된 단축명을 가진 지표의 핸들을 반환	차트 작업
ChartIndicatorName	지정된 차트 창의 지표 목록에 있는 수를 기준으로 지표의 단축명을 반환	차트 작업
ChartIndicatorsTotal	지정된 차트 창에 적용된 모든 지표의 수를 반환	차트 작업

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
ChartNavigate	차트에서 지정된 포지션을 기준으로 지정된 막대 수만큼 지정된 차트의 이동을 수행	차트 작업
ChartNext	지정한 차트 옆에 있는 차트의 차트 ID를 반환	차트 작업
ChartOpen	지정된 심볼 및 기간으로 새 차트를 열기	차트 작업
CharToString	심볼 코드를 한 문자 문자열로 변환	변환 함수
ChartPeriod	지정된 차트의 기간 값을 반환	차트 작업
ChartPriceOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트로 가격 좌표를 반환	차트 작업
ChartRedraw	지정된 차트의 강제 재작성 요청	차트 작업
ChartSaveTemplate	지정된 이름의 템플릿에 현재 차트 설정을 저장	차트 작업
ChartScreenShot	지정된 확장자에 따라 GIF, PNG 또는 BMP 형식의 현재 상태 차트를 스크린샷을 제공	차트 작업
ChartSetDouble	지정된 차트의 해당 속성에 대한 double 값을 설정	차트 작업
ChartSetInteger	지정된 차트의 해당 속성에 대한 정수 값(datetime, int, color, bool or char)을 설정	차트 작업
ChartSetString	지정된 차트의 해당 속성에 대한 string 값을 설정	차트 작업
ChartSetSymbolPeriod	지정된 차트의 심볼 값과 기간을 변경	차트 작업
ChartSymbol	지정된 차트의 심볼명 반환	차트 작업
ChartTimeOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트로 시간 좌표를 반환	차트 작업
ChartTimePriceToXY	시간/가격 표시에서 X 및 Y 좌표로 차트의 좌표를 변환	차트 작업
ChartWindowFind	지표가 그려진 하위창의 숫자를 반환	차트 작업
ChartWindowOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 하위창의 수	차트 작업

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
	(인덱스)를 반환	
ChartXOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트의 X 좌표를 반환	차트 작업
ChartXYToTimePrice	차트의 X 및 Y 좌표를 시간 및 가격 값으로 변환	차트 작업
ChartYOnDropped	엑스퍼트 어드바이저 또는 스크립트가 떨어진 차트 포인트의 Y 좌표를 반환	차트 작업
CheckPointer	개체 포인터의 유형을 반환	공통 함수
CLBufferCreate	OpenCL 버퍼 생성	OpenCL 사용
CLBufferFree	OpenCL 버퍼 삭제	OpenCL 사용
CLBufferRead	배열로 OpenCL 버퍼 읽기	OpenCL 사용
CLBufferWrite	OpenCL 버퍼로 배열 작성	OpenCL 사용
CLContextCreate	OpenCL 컨텍스트 생성	OpenCL 사용
CLContextFree	OpenCL 컨텍스트 제거	OpenCL 사용
CLExecute	OpenCL 프로그램 실행	OpenCL 사용
CLGetDeviceInfo	OpenCL 드라이버로부터 장치 속성 수신	OpenCL 사용
CLGetInfoInteger	OpenCL 개체 또는 장치의 정수 속성 값 반환	OpenCL 사용
CLHandleType	OpenCL 핸들 유형을 ENUM_OPENCL_HANDLE_TYPE 열거값으로 반환	OpenCL 사용
CLKernelCreate	OpenCL 시작 기능 생성	OpenCL 사용
CLKernelFree	OpenCL 시작 기능 제거	OpenCL 사용
CLProgramCreate	소스코드에서 OpenCL 프로그램 생성	OpenCL 사용
CLProgramFree	OpenCL 프로그램 제거	OpenCL 사용
CLSetKernelArg	OpenCL 기능을 위한 매개변수 설정	OpenCL 사용
CLSetKernelArgMem	OpenCL 기능의 매개변수로 OpenCL 버퍼 설정	OpenCL 사용
ColorToARGB	색상의 ARGB 표현을 수신하기 위해 색 유형을 uint 유형으로 변환.	변환 함수

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
ColorToString	"R,G,B" 문자열로 색상 값을 변환	변환 함수
코멘트	차트의 왼쪽 상단 코너에 코멘트를 출력	공통 함수
CopyBuffer	지정된 지표에서 배열로 지정된 버퍼의 데이터를 가져오기	시계열 및 지표 액세스
CopyClose	지정된 심볼 및 기간에 대한 막대 마감 가격에 대한 기록 데이터를 배열로 가져오기	시계열 및 지표 액세스
CopyHigh	지정된 심볼 및 기간에 대한 최대 막대 가격에 대한 기록 데이터를 배열로 가져오기	시계열 및 지표 액세스
CopyLow	지정된 심볼 및 기간에 대한 최소 막대 가격에 대한 내역 데이터를 배열로 가져오기	시계열 및 지표 액세스
CopyOpen	지정된 심볼 및 기간에 대한 막대 시작부분 가격에 대한 내역 데이터를 배열로 가져오기	시계열 및 지표 액세스
CopyRates	지정된 기호와 기간에 대한 요율 구조의 내역 데이터를 가져오기	시계열 및 지표 액세스
CopyRealVolume	지정된 심볼 및 기간의 거래 볼륨에 대한 내역 데이터를 배열로 가져오기	시계열 및 지표 액세스
CopySpread	지정된 기호 및 기간에 대한 스프레드의 내역 데이터를 배열로 가져오기	시계열 및 지표 액세스
CopyTicks	현재 작업 세션의 터미널에서 누적된 틱을 배열로 가져오기	시계열 및 지표 액세스
CopyTickVolume	지정된 기호 및 기간에 대한 틱 볼륨의 내역 데이터를 배열로 가져오기	시계열 및 지표 액세스
CopyTime	지정된 심볼 및 기간에 대한 막대 시작부분 시간의 내역 데이터를 배열로 가져오기	시계열 및 지표 액세스
cos	수의 코사인을 반환	산술 함수
CryptDecode	배열에서 데이터의 역변환을 수행	공통 함수
CryptEncode	지정된 방법을 사용하여 배열에서 데이터를 변환	공통 함수

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
CustomSymbolCreate	지정된 그룹에 지정된 이름을 가진 사용자 지정 심볼을 생성	사용자 지정 심볼
CustomSymbolDelete	지정된 이름의 사용자 지정 심볼 삭제	사용자 지정 심볼
CustomSymbolSetInteger	사용자 지정 심볼에 대한 정수 유형 속성 값 설정	사용자 지정 심볼
CustomSymbolSetDouble	사용자 지정 심볼에 대한 실제 유형 속성 값 설정	사용자 지정 심볼
CustomSymbolSetString	사용자 지정 심볼에 대한 문자열 유형 속성 값 설정	사용자 지정 심볼
CustomSymbolSetMarginRate	사용자 지정 심볼에 대한 주문 유형 및 방향에 따른 마진율 설정	사용자 지정 심볼
CustomSymbolSetSessionQuote	지정된 심볼 및 평일에 대해 지정된 시세 세션의 시작 및 종료 시간 설정	사용자 지정 심볼
CustomSymbolSetSessionTrade	지정된 심볼 및 평일에 대해 지정된 트레이딩 세션의 시작 및 종료 시간 설정	사용자 지정 심볼
CustomRatesDelete	지정된 시간 간격의 사용자 지정 심볼 가격 기록에서 모든 막대 삭제	사용자 지정 심볼
CustomRatesReplace	지정된 시간 간격 내에 사용자 지정 심볼의 가격 기록을 MqlRates 유형 배열의 대열로 완전히 교체	사용자 지정 심볼
CustomRatesUpdate	사용자 지정 심볼 기록에 결손 막대를 추가하고 기존 데이터를 MqlRates 유형 배열의 막대로 교체	사용자 지정 심볼
CustomTicksAdd	MqlTick 유형의 배열에서 사용자 지정 심볼의 가격 기록에 데이터를 추가 Market Watch 창에서 사용자 지정 심볼을 선택해야 함	사용자 지정 심볼
CustomTicksDelete	지정된 시간 간격의 사용자 지정 심볼 가격 기록에서 모든 틱 삭제	사용자 지정 심볼
CustomTicksReplace	지정된 시간 간격 내에 사용자 지정 심볼의 가격 기록을 MqlTick 유형 배열의 데이터로 완전히 교체	사용자 지정 심볼
CustomBookAdd	다음을 전달 – 사용자 지정 심볼에 대한 마켓의 뎁스(Depth of	사용자 지정 심볼

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
	Market)의 상태	
DatabaseOpen	지정된 파일에 데이터베이스를 열거나 생성	데이터베이스 작업
DatabaseClose	데이터베이스 닫기	데이터베이스 작업
DatabaseImport	파일에서 테이블로 데이터 가져오기	데이터베이스 작업
DatabaseExport	테이블 또는 SQL 요청 실행 결과를 CSV 파일로 내보내기	데이터베이스 작업
DatabasePrint	테이블 또는 SQL 요청 실행 결과를 엑스퍼터 저널에 인쇄	데이터베이스 작업
DatabaseTableExists	데이터베이스에 테이블이 있는지 확인	데이터베이스 작업
DatabaseExecute	지정된 데이터베이스에 대한 요청을 실행	데이터베이스 작업
DatabasePrepare	DatabaseRead()를 사용하여 실행할 수 있는 요청의 핸들 생성	데이터베이스 작업
DatabaseReset	다음을 불러온 후, 요청을 재설정 -DatabasePrepare()	데이터베이스 작업
DatabaseBind	요청에 매개변수 값을 설정	데이터베이스 작업
DatabaseBindArray	배열을 변수 값으로 설정	데이터베이스 작업
DatabaseRead	요청 결과로 다음 항목으로 이동	데이터베이스 작업
DatabaseReadBind	다음 기록으로 이동하고 기록에서 구조로 데이터를 읽기	데이터베이스 작업
DatabaseFinalize	DatabasePrepare()에서 작성된 요청을 제거	데이터베이스 작업
DatabaseTransactionBegin	트랜잭션 실행 시작	데이터베이스 작업
DatabaseTransactionCommit	트랜잭션 실행 완료	데이터베이스 작업
DatabaseTransactionRollback	트랜잭션 롤 백	데이터베이스 작업
DatabaseColumnsCount	요청에서 필드의 수를 가져오기	데이터베이스 작업
DatabaseColumnName	인덱스별로 필드 이름 가져오기	데이터베이스 작업
DatabaseColumnType	인덱스별로 필드 유형 가져오기	데이터베이스 작업
DatabaseColumnSize	필드 크기(바이트) 가져오기	데이터베이스 작업
DatabaseColumnText	현재 기록에서 문자열로 필드값 가져오기	데이터베이스 작업

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
DatabaseColumnInteger	현재 기록에서 int 유형 값 가져오기	데이터베이스 작업
DatabaseColumnLong	현재 기록에서 long 유형 값 가져오기	데이터베이스 작업
DatabaseColumnDouble	현재 레코드에서 double 유형 값 가져오기	데이터베이스 작업
DatabaseColumnBlob	현재 기록에서 필드 값을 배열로 가져오기	데이터베이스 작업
DebugBreak	디버깅의 중지포인트 프로그램	공통 함수
Digits	현재 차트 심볼 가격 값의 정확도를 결정하는 소수 자릿수를 반환	검사
DoubleToString	수의 값을 지정된 정확도로 텍스트 라인 변환	변환 함수
DXContextCreate	지정된 크기의 프레임을 렌더링하기 위한 그래픽 컨텍스트를 생성	DirectX 작업
DXContextSetSize	DXContextCreate()에서 작성된 그래픽 컨텍스트의 프레임 크기를 변경	DirectX 작업
DXContextSetSize	DXContextCreate()에서 생성된 그래픽 컨텍스트의 프레임 크기를 가져오기	DirectX 작업
DXContextClearColors	렌더링 버퍼에 대해 지정된 색상을 모든 픽셀로 설정	DirectX 작업
DXContextClearDepth	깊이 버퍼를 비움	DirectX 작업
DXContextGetColors	그래픽 컨텍스트에서 지정된 크기 및 오프셋의 이미지를 가져오기	DirectX 작업
DXContextGetDepth	렌더링된 프레임의 깊이 버퍼를 가져오기	DirectX 작업
DXBufferCreate	데이터 배열을 기반으로 지정된 유형의 버퍼를 생성	DirectX 작업
DXTextureCreate	전달된 이미지에서 잘라낸 지정된 크기의 직사각형으로 2D 질감 생성	DirectX 작업
DXInputCreate	셰이더 입력 생성	DirectX 작업
DXInputSet	셰이더 입력 설정	DirectX 작업
DXShaderCreate	지정된 유형의 셰이더 생성	DirectX 작업

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
DXShaderSetLayout	정점 셰이더의 꼭지점 레이아웃을 설정	DirectX 작업
DXShaderInputsSet	셰이더 입력 설정	DirectX 작업
DXShaderTexturesSet	셰이더 질감 설정	DirectX 작업
DXDraw	DXBufferSet()에서 정점 버퍼 세트의 점들을 렌더링	DirectX 작업
DXDrawIndexed	DXBufferSet()에서 인덱스 버퍼가 설명하는 그래픽 기본 요소를 렌더링	DirectX 작업
DXPrimiveTopologySet	DXDrawIndexed()를 사용하여 렌더링할 기본 요소의 유형을 설정	DirectX 작업
DXBufferSet	현재 렌더링의 버퍼를 설정	DirectX 작업
DXShaderSet	렌더링을 위한 셰이더 설정	DirectX 작업
DXHandleType	핸들 유형을 반환	DirectX 작업
DXRelease	핸들을 해제	DirectX 작업
EnumToString	모든 유형의 열거값을 문자열로 변환	변환 함수
EventChartCustom	지정된 차트에 대한 사용자 지정 이벤트를 성성	이벤트 작업
EventKillTimer	현재 차트에서 타이머에 의한 이벤트 생성을 중지	이벤트 작업
EventSetMillisecondTimer	현재 차트에 대해 1초 미만의 주기를 가진 고해상도 타이머의 이벤트 생성기를 시작	이벤트 작업
EventSetTimer	현재 차트에 대해 지정된 주기로 타이머 이벤트 생성기를 시작	이벤트 작업
exp	숫자의 지수를 반환	산술 함수
ExpertRemove	엑스퍼트 어드바이저를 중지하고 차트에서 언로드	공통 함수
fabs	지정된 숫자 값의 절대값(계수)를 반환	산술 함수
FileClose	이전에 연 파일을 닫기	파일 기능
FileCopy	로컬 또는 공유 폴더에서 다른 파일로 원본 파일 복사	파일 기능
FileDelete	지정된 파일을 삭제	파일 기능
FileFindClose	검색 핸들을 닫기	파일 기능

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
FileFindFirst	지정된 필터에 따라 디렉토리에서 파일 검색을 시작	파일 기능
FileFindNext	FileFindFirst() 기능으로 시작한 검색을 계속 진행	파일 기능
FileFlush	입력/출력 파일 버퍼에 남아있는 모든 데이터를 디스크에 작성	파일 기능
FileGetInteger	파일의 정수 속성을 가져오기	파일 기능
FileIsEnding	읽는 과정에서 파일의 끝을 정하기	파일 기능
FileIsExist	파일이 있는지 확인	파일 기능
FileIsLineEnding	읽는 과정 중 텍스트 파일에서 줄의 끝을 정하기	파일 기능
FileMove	파일 이동 또는 이름 변경	파일 기능
FileOpen	지정된 이름과 플래그가 있는 파일을 열기	파일 기능
FileReadArray	BIN 유형의 파일에서 문자열을 제외한 모든 유형의 배열을 읽기	파일 기능
FileReadBool	CSV의 파일에서 현재 위치에서 구분 기호(또는 텍스트 줄의 끝)까지 문자열을 입력하고읽고 문자열을 bool 유형 갑으로 변환	파일 기능
FileReadDatetime	CSV 파일에서 다음 형식중 하나의 문자열을 읽습니다: "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" 또는 "HH:MM:SS" – 이 값을 날짜시간 값으로 변환	파일 기능
FileReadDouble	파일 포인터의 현재 포지션에서 이중 값을 읽기	파일 기능
FileReadFloat	파일 포인터의 현재 포지션에서 플로트(float) 값을 읽기	파일 기능
FileReadInteger	파일 포인터의 현재 위치에서 int, short 또는 char 값을 읽기	파일 기능
FileReadLong	파일 포인터의 현재 포지션에서 long 유형 값 읽기	파일 기능
FileReadNumber	CSV 파일의 현재 포지션에서 구분 기호(또는 텍스트 줄의 끝)까지 문자열을 읽기 및 문자열을 이중값으로 변환	파일 기능

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
FileReadString	파일에서 파일 포인터의 현재 포지션의 문자열 읽기	파일 기능
FileReadStruct	바이너리 파일의 내용을 파일 포인터의 현재 포지션에서 매개변수로 전달된 구조로 읽기	파일 기능
FileSeek	파일 포인터의 포지션을 지정된 바이트 수만큼 지정된 포지션으로 이동	파일 기능
FileSize	해당 열기 파일의 크기를 반환	파일 기능
FileTell	해당 열기 파일의 파일 포인터의 현재 포지션을 반환	파일 기능
FileWrite	CSV 또는 TXT 유형의 파일에 데이터 쓰기	파일 기능
FileWriteArray	문자열을 제외한 모든 유형의 배열을 BIN 유형 파일에 쓰기	파일 기능
FileWriteDouble	파일 포인터의 현재 포지션에서 바이너리 파일에 double 유형의 값을 쓰기	파일 기능
FileWriteFloat	파일 포인터의 현재 포지션에서 바이너리 파일에 float 유형의 값 쓰기	파일 기능
FileWriteInteger	파일 포인터의 현재 포지션에서 바이너리 파일에 int 유형의 값 쓰기	파일 기능
FileWriteLong	파일 포인터의 현재 포지션에서 바이너리 파일에 long 유형의 값 쓰기	파일 기능
FileWriteString	파일 포인터의 현재 포지션에서 시작하여 BIN 또는 TXT 파일에 문자열 매개변수 값을 쓰기	파일 기능
FileWriteStruct	파일 포인터에서 현재 포지션에서 시작하여 매개변수로 전달된 구조의 내용을 바이너리 파일에 쓰기	파일 기능
floor	아래에서 가장 가까운 정수 숫자 값을 반환	산술 함수
fmax	두 숫자 값의 최대값을 반환	산술 함수
fmin	두 숫자 값의 최소값을 반환	산술 함수

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
fmod	두 숫자의 나눗셈 후 실제 나머지를 반환	산술 함수
FolderClean	지정된 폴더의 모든 파일을 삭제	파일 기능
FolderCreate	Files 디렉토리에 폴더를 생성	파일 기능
FolderDelete	선택된 디렉토리를 제거 폴더가 비어있지 않으면 제거할 수 없습니다.	파일 기능
FrameAdd	데이터가 있는 프레임을 추가	최적화 결과 작업
FrameFilter	프레임 읽기 필터를 설정하고 포인터를 시작 부분으로 이동	최적화 결과 작업
FrameFirst	프레임 판독의 포인터를 시작 부분으로 이동하고 이전에 설정된 필터를 재설정	최적화 결과 작업
FrameInputs	수신하는 입력 매개변수는 프레임이 형성됨	최적화 결과 작업
FrameNext	프레임을 읽고 포인터를 다음 프레임으로 이동	최적화 결과 작업
GetLastError	마지막 오류 반환	검사
GetPointer	반환하는 개체포인터	공통 함수
GetTickCount	시스템이 시작된 후 경과된 밀리초 수를 반환	공통 함수
GlobalVariableCheck	지정된 이름의 글로벌 변수가 있는지 확인	터미널의 글로벌 변수
GlobalVariableDel	글로벌 변수 삭제	터미널의 글로벌 변수
GlobalVariableGet	글로벌 변수 값 반환	터미널의 글로벌 변수
GlobalVariableName	글로벌 변수 목록에서 글로벌 변수의 이름을 순서 번호별로 반환	터미널의 글로벌 변수
GlobalVariablesDeleteAll	이름에 지정된 접두사가 있는 글로벌 변수를 삭제	터미널의 글로벌 변수
GlobalVariableSet	새 값을 글로벌 변수로 설정	터미널의 글로벌 변수
GlobalVariableSetOnCondition	조건별로 기존 글로벌 변수의 새 값을 설정	터미널의 글로벌 변수
GlobalVariablesFlush	모든 글로벌 변수의 내용을 디스크에 강제 저장	터미널의 글로벌 변수
GlobalVariablesTotal	글로벌 변수의 전체 수를 반환	터미널의 글로벌 변수

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
GlobalVariableTemp	터미널의 현재 세션에만 존재하는 글로벌 변수로 새 값을 설정	터미널의 글로벌 변수
GlobalVariableTime	글로벌 변수에 마지막으로 액세스한 시간을 반환	터미널의 글로벌 변수
HistoryDealGetDouble	내역에서 거래 요청된 속성(double)을 반환	트레이드 기능
HistoryDealGetInteger	내역에서 거래의 요청된 속성(날짜시간 또는 int)을 반환	트레이드 기능
HistoryDealGetString	내역에서 거래의 요청된 속성(string)을 반환	트레이드 기능
HistoryDealGetTicket	내역에 해당하는 거래의 티켓을 반환	트레이드 기능
HistoryDealSelect	적절한 기능을 통해 거래를 추가로 요청하기 위해 내역에서 거래를 선택	트레이드 기능
HistoryDealsTotal	내역의 거래 수를 반환	트레이드 기능
HistoryOrderGetDouble	내역에서 주문의 요청된 속성(double)을 반환	트레이드 기능
HistoryOrderGetInteger	내역에서 주문의 요청된 속성(날짜시간 또는 int)을 반환	트레이드 기능
HistoryOrderGetString	내역에서 주문의 요청된 속성(string)을 반환	트레이드 기능
HistoryOrderGetTicket	내역에서 해당 주문의 주문 티켓을 반환	트레이드 기능
HistoryOrderSelect	추가로 작업하기 위해 내역에서 주문을 선택	트레이드 기능
HistoryOrdersTotal	내역에서 주문 수를 반환	트레이드 기능
HistorySelect	서버 시간의 지정된 기간에 대한 트랜잭션 및 주문 기록을 검색	트레이드 기능
HistorySelectByPosition	지정된 포지션 지표를 사용한 거래 내역 요청.	트레이드 기능
iBars	내역에서 사용가능한 해당 심볼 및 기간의 막대 수를 반환	시계열 및 지표 액세스
iBarShift	지정된 시간에 해당하는 막대의 인덱스를 반환	시계열 및 지표 액세스
iClose	해당 차트에서 막대('shift' 매개변수로 표시됨)의 마감 가격을 반환	시계열 및 지표 액세스

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
iHigh	해당 차트에서 바('shift' 매개변수로 표시됨)의 고가를 반환	시계열 및 지표 액세스
iHighest	해당 차트에 있는 최고 값(현재 막대에 상대적 시프트)의 인덱스를 반환	시계열 및 지표 액세스
iLow	해당 차트에서 바('shift' 매개변수로 표시됨)의 저가를 반환	시계열 및 지표 액세스
iLowest	해당 차트에 있는 최소 값(현재 막대에 상대적 시프트)의 인덱스를 반환	시계열 및 지표 액세스
iOpen	해당 차트에서 막대('shit' 매개변수로 표시됨)의 시작 가격을 반환	시계열 및 지표 액세스
iTime	해당 차트에서 막대('shit' 매개변수로 표시됨)의 시작 시간을 반환	시계열 및 지표 액세스
iTickVolume	해당 차트에서 막대('shift' 매개변수로 표시됨)의 틱 볼륨을 반환	시계열 및 지표 액세스
iRealVolume	해당 차트에서 막대('shift' 매개변수로 표시됨)의 실제 볼륨을 반환	시계열 및 지표 액세스
iVolume	해당 차트에서 막대('shift' 매개변수로 표시됨)의 틱 볼륨을 반환	시계열 및 지표 액세스
iSpread	해당 차트에서 막대('shift' 매개변수로 표시됨)의 스프레드 값을 반환	시계열 및 지표 액세스
iAD	누적/분배	기술 지표
iADX	평균 방향 인덱스	기술 지표
iADXWilder	웰즈 와일더의 평균 방향 인덱스	기술 지표
iAlligator	엘리게이터	기술 지표
iAMA	적응형 이동 평균	기술 지표
iAO	어썸 오실레이터	기술 지표
iATR	평균 참값 범위	기술 지표
iBands	볼링어 밴즈®	기술 지표
iBearsPower	베어즈 파워	기술 지표
iBullsPower	불스 파워	기술 지표
iBWMFI	빌 윌리엄즈의 시장 촉진 인덱스	기술 지표
iCCI	상품 채널 인덱스	기술 지표

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
iChaikin	체이킨 오실레이터	기술 지표
iCustom	고객 지표	기술 지표
iDEMA	이중 지수 이동 평균	기술 지표
iDeMarker	DeMarker	기술 지표
iEnvelopes	봉투	기술 지표
iForce	포스 인덱스	기술 지표
iFractals	프랙탈	기술 지표
iFrAMA	프랙탈 적응형 이동 평균	기술 지표
iGator	게이터 오실레이터	기술 지표
iIchimoku	이치모쿠 킨코 효	기술 지표
iMA	이동 평균	기술 지표
iMACD	이동 평균 수렴-분산	기술 지표
iMFI	자금 흐름 인덱스	기술 지표
iMomentum	모멘텀	기술 지표
IndicatorCreate	다음에서 생성된 지정 기술 지표로 핸들 반환: MqlParam 유형 매개변수	시계열 및 지표 액세스
IndicatorParameters	지정된 핸들을 기준으로, 지표의 입력 매개변수 숫자 뿐만 아니라 변수의 값 및 유형을 반환	시계열 및 지표 액세스
IndicatorRelease	다른 사용자가 사용하지 않는 경우 지표 핸들을 제거하거 지표의 계산 블록을 해제	시계열 및 지표 액세스
IndicatorSetDouble	다음 유형의 지표 속성 값을 설정 : double 유형	고객 지표
IndicatorSetInteger	다음 유형의 지표 속성 값을 설정 : int 유형	고객 지표
IndicatorSetString	다음 유형의 지표 속성 값을 설정 : string 유형	고객 지표
IntegerToString	사전 설정된 길의 문자열로 변환	변환 함수
iOBV	잔고량	기술 지표
iOsMA	오실레이터의 이동 평균 (MACD 히스토그램)	기술 지표
iRSI	상대 강도 지수	기술 지표

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
iRVI	상대 활력 지수	기술 지표
iSAR	포물선 정지 및 역 시스템	기술 지표
IsStopped	mql5 프로그램이 작동을 중지하도록 명령을 받은 경우 참값을 반환	검사
iStdDev	표준 편차	기술 지표
iStochastic	스토캐스틱 오실레이터	기술 지표
iTEMA	삼중 지수 이동 평균	기술 지표
iTriX	삼중 지수 이동 평균 오실레이터	기술 지표
iVIDyA	변수 인덱스 동적 평균	기술 지표
iVolumes	볼륨	기술 지표
iWPR	윌리엄즈의 퍼센트 범위	기술 지표
log	자연 로그 반환	산술 함수
log10	상용 로그의 수를 반환	산술 함수
MarketBookAdd	선택된 심볼에 대한 마켓의 깊이(Depth of Market) 열기를 제공하고 DOM 변경 알림을 수신하기 위해 구독.	마켓 정보
MarketBookGet	지정된 기호의 마켓 뎁스 기록이 포함된 MqlBookInfo배열 구조를 반환	마켓 정보
MarketBookRelease	선택된 심볼에 대한 마켓의 깊이(Depth of Market) 닫기를 제공하고 DOM 변경 알림을 수신하기 위해 구독을 취소	마켓 정보
MathAbs	지정된 숫자 값의 절대값(계수)를 반환	산술 함수
MathArccos	라디안 단위 x의 아크 코사인을 반환	산술 함수
MathArcsin	라디안 단위로 x의 아크 사인을 반환	산술 함수
MathArctan	라디안 단위의 x의 아크 탄젠트를 반환	산술 함수
MathCeil	위에서 가장 가까운 정수 숫자 값을 반환	산술 함수
MathCos	수의 코사인을 반환	산술 함수

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
MathExp	숫자의 지수를 반환	산술 함수
MathFloor	아래에서 가장 가까운 정수 숫자 값을 반환	산술 함수
MathIsValidNumber	실수의 정확성을 확인	산술 함수
MathLog	자연 로그 반환	산술 함수
MathLog10	상용 로그의 수를 반환	산술 함수
MathMax	두 숫자 값의 최대값을 반환	산술 함수
MathMin	두 숫자 값의 최소값을 반환	산술 함수
MathMod	두 숫자의 나눗셈 후 실제 나머지를 반환	산술 함수
MathPow	베이스를 지정된 파워로 올림	산술 함수
MathRand	0 부터 32767까지 범위 내 유사 랜덤 값을 반환	산술 함수
MathRound	가장 가까운 정수로 값 반올림	산술 함수
MathSin	숫자의 사인을 반환	산술 함수
MathSqrt	제곱근을 반환	산술 함수
MathSrand	일련의 유사 랜덤 정수를 생성하기 위한 시작점을 설정	산술 함수
MathTan	수의 탄젠트를 반환	산술 함수
MessageBox	메시지 상자를 만들고 표시 및 관리	공통 함수
MQLInfoInteger	실행 중인 mql5 프로그램의 해당 속성의 정수 값을 반환	검사
MQLInfoString	실행 중인 mql5 프로그램의 해당 속성의 문자열 값을 반환	검사
MT5Initialize	MetaTrader 5 터미널과 연결을 설정	파이톤(Python)용 MetaTrader
MT5Shutdown	MetaTrader 5 터미널에 대한 이전에 설정된 연결을 닫기	파이톤(Python)용 MetaTrader
MT5TerminalInfo	연결된 MetaTrader 5 터미널의 상태 및 매개변수 가져오기	파이톤(Python)용 MetaTrader
MT5Version	MetaTrader 5 터미널 버전 반환	파이톤(Python)용 MetaTrader
MT5CopyRatesFrom	지정된 날짜부터 시작하는 MetaTrader 5 터미널에서 막대 가져오기	파이톤(Python)용 MetaTrader

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
MT5CopyRatesFromPos	지정된 인덱스에서 시작하는 MetaTrader 5 터미널에서 막대 가져오기	파이톤(Python)용 MetaTrader
MT5CopyRatesRange	MetaTrader 5 터미널에서 지정된 날짜 범위의 막대 가져오기	파이톤(Python)용 MetaTrader
MT5CopyTicksFrom	지정된 날짜부터 시작하는 MetaTrader 5 터미널에서 틱 가져오기	파이톤(Python)용 MetaTrader
MT5CopyTicksRange	MetaTrader 5 터미널에서 지정된 날짜 범위에 대한 틱 가져오기	파이톤(Python)용 MetaTrader
NormalizeDouble	부동 소수점 숫자의 지정된 정확도에 대한 반올림	변환 함수
ObjectCreate	지정된 타츠에 지정된 유형의 개체를 생성	오브젝트 기능
ObjectDelete	지정된 차트(지정된 차트 하위 창에서)에서 지정된 이름의 개체를 제거	오브젝트 기능
ObjectFind	이름으로 지정된 ID를 가진 개체를 검색	오브젝트 기능
ObjectGetDouble	해당 개체 속성의 double 값을 반환	오브젝트 기능
ObjectGetInteger	해당 개체 속성의 정수 값을 반환	오브젝트 기능
ObjectGetString	해당 개체 속성의 string 값을 반환	오브젝트 기능
ObjectGetTimeByValue	지정된 개체 가격 값에 대한 시간 값을 반환	오브젝트 기능
ObjectGetValueByTime	지정된 시간 동안 개체의 가격 값을 반환	오브젝트 기능
ObjectMove	지정된 개체 고정점의 좌표를 변경	오브젝트 기능
ObjectName	지정된 차트(지정된 차트 하위창)에서 해당 유형의 개체 이름을 반환	오브젝트 기능
ObjectsDeleteAll	지정된 차트(지정된 차트 하위 창에서)에서 지정된 유형의 모든 개체를 제거	오브젝트 기능
ObjectSetDouble	해당 개체 속성의 값을 설정	오브젝트 기능
ObjectSetInteger	해당 개체 속성의 값을 설정	오브젝트 기능

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
ObjectSetString	해당 개체 속성의 값을 설정	오브젝트 기능
ObjectsTotal	지정된 차트(지정된 차트 하위창)에서 지정된 유형의 개체 수를 반환	오브젝트 기능
OnStart	이 기능은 스크립트에 설정된 작업을 수행하기 위해 시작 이벤트가 발생하면 호출됩니다	이벤트 핸들링
OnInit	이 기능은 MQL5 프로그램을 초기화하기 위해 Init 이벤트가 발생하면 지표 및 EA에 호출됩니다	이벤트 핸들링
OnDeinit	이 기능은 시작된 MQL5 프로그램을 초기화 하기 위해 Deinit 이벤트가 발생하면 지표 및 EA에 호출됩니다	이벤트 핸들링
OnTick	이 기능은 새 시세를 핸들링하기 위해 NewTick 이벤트가 발생하면 EA에 호출됩니다	이벤트 핸들링
OnCalculate	이 기능은 가격 데이터 변경을 처리하기 위해 계산 이벤트가 발생하면 지표에 호출됩니다	이벤트 핸들링
OnTimer	이 기능은 고정된 시간 간격으로 터미널에 의해 생성된 타이머 주기 이벤트동안 지표와 EA에 호출됩니다	이벤트 핸들링
OnTrade	이 기능은 거래 서버에서 트레이딩 작업이 끝날 때 생성된 트레이드 이벤트 중에 EA에 호출됩니다	이벤트 핸들링
OnTradeTransaction	이 기능은 거래 요청 실행 결과를 처리하기 위해 거래 트랜잭션 이벤트가 발생할 때 EA에 호출됩니다	이벤트 핸들링
OnBookEvent	이 기능은 시장 깊이 변화를 처리하기 위해 BookEvent 이벤트가 발생하면 EA에 호출됩니다	이벤트 핸들링
OnChartEvent	이 기능은 사용자 또는 MQL5 프로그램에 의해 수행된 차트 변경을 처리 하기 위해 ChartEvent 이벤트가 발생하면 지표 및 EA에 호출됩니다	이벤트 핸들링
OnTester	이 기능은 내역 데이터에 대해 EA를 테스트를 한 후 테스터 이벤트	이벤트 핸들링

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
	가 발생하면 EA에 호출됩니다	
OnTesterInit	이 기능은 전략 테스터에서 최적화하기 전에 필요한 작업을 수행하기 위해 TesterInit 이벤트가 발생하면 EA에 호출됩니다	이벤트 핸들링
OnTesterDeinit	이 기능은 전략 테스터에서 EA 최적화 후 TesterDeinit 이벤트가 발생하면 EA에 호출됩니다	이벤트 핸들링
OnTesterPass	이 기능은 전략 테스터에서 EA 최적화 중 새 데이터 프레임 도착을 처리하기 위해 TesterPass 이벤트가 발생하면 EA에 호출됩니다	이벤트 핸들링
OrderCalcMargin	지정된 주문 유형에 필요한 마진을 예금 통화로 계산	트레이드 기능
OrderCalcProfit	전달된 매개변수를 기준으로 예금 통화로 수익을 계산	트레이드 기능
OrderCheck	다음의 필요한 내용을 실행할 수 있는 자금이 충분한지 확인: 거래 수행.	트레이드 기능
OrderGetDouble	주문의 요청된 속성(double)을 반환	트레이드 기능
OrderGetInteger	주문의 요청된 속성(날짜시간 또는 int)을 반환	트레이드 기능
OrderGetString	주문의 요청된 속성(string)을 반환	트레이드 기능
OrderGetTicket	해당 주문의 티켓을 반환	트레이드 기능
OrderSelect	추가로 작업할 주문을 선택	트레이드 기능
OrderSend	서버로 거래 요청 전송	트레이드 기능
OrderSendAsync	비동기식으로 거래 요청을 거래 서버의 거래 응답을 기다리지 않고 전송	트레이드 기능
OrdersTotal	주문 수를 반환	트레이드 기능
ParameterGetRange	전략 테스터에서 엑스퍼트 어드바이저를 최적화 할 때 입력 변수의 단계에 변경 및 값의 범위 데이터를 수신	최적화 결과 작업
ParameterSetRange	전략 테스터에서 엑스퍼트 어드바이저를 최적화 할 때 입력 변수	최적화 결과 작업

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
	사용(값, 변경 단계, 초기 및 최종 값)을 지정	
Period	현재 차트 타임프레임을 반환	검사
PeriodSeconds	기간 내 초 수를 반환	공통 함수
PlaySound	사운드 파일을 재생	공통 함수
PlotIndexGetInteger	다음 유형의 지시선 속성 값을 반환: 정수 유형	고객 지표
PlotIndexSetDouble	다음 유형의 지시선 속성 값을 설정: double	고객 지표
PlotIndexSetInteger	다음 유형의 지시선 속성 값을 설정: int 유형	고객 지표
PlotIndexSetString	다음 유형의 지시선 속성 값을 설정: string 유형	고객 지표
Point	시세 통화에서 현재 심볼의 포인트 크기를 반환	검사
PositionGetDouble	개장 포지션의 요청된 속성(double)을 반환	트레이드 기능
PositionGetInteger	개장 포지션의 요청된 속성(날짜 시간 또는 int)을 반환	트레이드 기능
PositionGetString	개장 포지션의 요청된 속성(string)을 반환	트레이드 기능
PositionGetSymbol	개장 포지션에 해당하는 심볼을 반환	트레이드 기능
PositionGetTicket	개장 포지션 목록에 지정된 인덱스가 있는 포지션의 티켓을 반환	트레이드 기능
PositionSelect	추가로 작업하기 위해 시작 포지션을 선택	트레이드 기능
PositionSelectByTicket	지정된 티켓 번호로 작업할 포지션을 선택	트레이드 기능
PositionsTotal	개장 포지션 수를 반환	트레이드 기능
pow	베이스를 지정된 파워로 올림	산술 함수
Print	로그에 메시지 표시	공통 함수
PrintFormat	미리 설정된 형식에 따라 로그 파일 심볼 및 값 세트를 포맷 및 인쇄	공통 함수

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
rand	0 부터 32767까지 범위 내 유사 랜덤 값을 반환	산술 함수
ResetLastError	사전 결정된 변수 _LastError 값을 0으로 설정	공통 함수
ResourceCreate	데이터 세트 기반으로 이미지 리소스 생성	공통 함수
ResourceFree	삭제: 동적으로 생성된 리소스 (해당 리소스에 할당된 메모리 사용 가능)	공통 함수
ResourceReadImage	다음의 그래픽 리소스에서 데이터 읽기: ResourceCreate() 기능으로 생성 또는 컴파일 중EX5 파일에 저장	공통 함수
ResourceSave	지정된 파일에 리소스를 저장	공통 함수
round	가장 가까운 정수로 값 반올림	산술 함수
SendFTP	"FTP" 탭의 설정 창에 지정된 주소로 파일 전송	공통 함수
SendMail	"Email" 탭의 설정 창에 지정된 주소로 이메일을 전송	공통 함수
SendNotification	"Notifications" 탭에서 MetaQuotes ID가 지정된 모바일 터미널로 푸쉬 알림 전송	공통 함수
SeriesInfoInteger	내역 데이터의 상태에 대한 정보를 반환	시계열 및 지표 액세스
SetIndexBuffer	지정된 지표 버퍼를 다음 유형의 1차원 동적 배열 로 바인딩: double 유형	고객 지표
ShortArrayToString	배열 부분을 문자열로 복사	변환 함수
ShortToString	심볼 코드(유니코드)를 하나의 심볼 문자열로 변환	변환 함수
SignalBaseGetDouble	선택된 시그널에 대한 double 유형 속성 값을 반환	거래 시그널
SignalBaseGetInteger	선택된 시그널에 대한 정수 유형 속성 값을 반환	거래 시그널
SignalBaseGetString	선택된 시그널에 대한 string 유형 속성 값을 반환	거래 시그널
SignalBaseSelect	추가로 작업하기 위해 터미널에서 이용가능한 시그널에서 시그	거래 시그널

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
	널을 선택	
SignalBaseTotal	터미널에서 이용가능한 총 시그널 양을 반환	거래 시그널
SignalInfoGetDouble	시그널 복사 설정의 double 유형 속성 값을 반환	거래 시그널
SignalInfoGetInteger	시그널 복사 설정의 정수 유형 속성 값을 반환	거래 시그널
SignalInfoGetString	시그널 복사 설정의 문자열 유형 속성 값을 반환	거래 시그널
SignalInfoSetDouble	시그널 복사 설정의 double 유형 속성 값을 설정	거래 시그널
SignalInfoSetInteger	시그널 복사 설정의 정수 유형 속성 값을 설정	거래 시그널
SignalSubscribe	트레이딩 시그널을 구독	거래 시그널
SignalUnsubscribe	구독 취소	거래 시그널
sin	숫자의 사인을 반환	산술 함수
Sleep	지정된 간격 내에 엑스퍼트 어드바이저 또는 스크립트의 실행을 일시 중단	공통 함수
SocketCreate	지정된 플래그가 있는 소켓을 생성하고 핸들을 반환	네트워크 기능
SocketClose	소켓 닫기	네트워크 기능
SocketConnect	시간 초과 제어를 사용하여 서버에 연결	네트워크 기능
SocketIsConnected	소켓이 현재 연결되어 있는지 확인	네트워크 기능
SocketIsReadable	소켓에서 읽을 수 있는 바이트 수 가져오기	네트워크 기능
SocketIsWritable	현재 소켓에서 데이터를 쓸 수 있는지를 확인	네트워크 기능
SocketTimeouts	소켓 시스템 개체에 대한 데이터 수신 및 전송 시간 초과 설정	네트워크 기능
SocketRead	소켓에서 데이터 읽기	네트워크 기능
SocketSend	소켓에 데이터 쓰기	네트워크 기능
SocketTlsHandshake	TLS 핸드셰이크 프로토콜을 통해 지정된 호스트에 대한 보안	네트워크 기능

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
	TLS(SSL) 연결 시작	
SocketTlsCertificate	보안 네트워크 연결에 사용되는 인증서 데이터 가져오기	네트워크 기능
SocketTlsRead	보안 TLS 연결에서 데이터 읽기	네트워크 기능
SocketTlsReadAvailable	보안 TLS 연결에서 사용가능한 모든 데이터 읽기	네트워크 기능
SocketTlsSend	Send data via secure TLS connection	네트워크 기능
sqrt	제곱근을 반환	산술 함수
srand	일련의 유사 랜덤 정수를 생성하기 위한 시작점을 설정	산술 함수
StringAdd	다른 문자열 끝에 문자열을 추가	문자열(String) 기능
StringBufferLen	문자열에 할당된 버퍼 크기를 반환	문자열(String) 기능
StringCompare	두 문자열을 비교하여 첫 번째 문자열이 두 번째 문자열보다 크면 1을 반환, 0은 문자열이 같은 경우, -1(마이너스1)은 첫 번째 문자열이 두 번째 문자열보다 작은 경우	문자열(String) 기능
StringConcatenate	통과된 변수 문자열을 형성	문자열(String) 기능
StringFill	지정된 문자열을 선택된 심볼로 채우기	문자열(String) 기능
StringFind	문자열에서 하위문자열 검색	문자열(String) 기능
StringFormat	사전 설정 형식에 따라 숫자를 문자열로 변환	변환 함수
StringGetCharacter	지정된 문자열 위치에 있는 수의 값을 반환	문자열(String) 기능
StringInit	지정된 심볼로 문자열을 초기화하고 지정된 문자열 길이를 제공	문자열(String) 기능
StringLen	문자열 심볼 수를 반환	문자열(String) 기능
StringReplace	문자열에서 발견한 모든 하위문자열을 일련의 기호로 교체	문자열(String) 기능
StringSetCharacter	지정된 포지션에 있는 심볼 값이 변경된 문자열의 복사본을 반환	문자열(String) 기능
StringSplit	얻은 문자열 수를 반환하는 지정된 문자열에서 지정된 구분기호	문자열(String) 기능

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
	를 사용하여 하위문자열을 가져오기	
StringSubstr	지정된 포지션에서 시작하는 텍스트 문자열에서 하위문자열을 추출	문자열(String) 기능
StringToCharArray	유니코드에서 ANSI로 변환된 문자열을 선택한 uchar 형식의 배열 위치로 기호별로 복사	변환 함수
StringToColor	색상 이름을 가진 "R,G,B" 문자열 또는 문자열을 색상 유형 값으로 변환	변환 함수
StringToDouble	숫자 심볼 표현을 포함하는 문자열을 double 유형의 수로 변환	변환 함수
StringToInteger	숫자 심볼 표현을 포함하는 문자열을 int 유형 수로 변환	변환 함수
StringToLower	선택된 문자열의 모든 심볼을 소문자로 변환	문자열(String) 기능
StringToShortArray	심볼에 따라 문자열을 ushort 유형의 배열에서 선택된 부분으로 복사	변환 함수
StringToTime	날짜 또는 시간을 포함하는 문자열을 "yyyy.mm.dd [hh:mi]" 형식으로 날짜 시간 유형으로 변환	변환 함수
StringToUpper	선택된 문자열의 모든 심볼을 대문자로 변환	문자열(String) 기능
StringTrimLeft	문자열의 왼쪽 부분에 있는 줄바꿈 문자, 공백 및 탭 잘라내기	문자열(String) 기능
StringTrimRight	문자열의 오른쪽 부분에 있는 줄바꿈 문자, 공백 및 탭 잘라내기	문자열(String) 기능
StructToTime	MqlDateTime 구조 유형의 변수를 날짜시간 값으로 변환₩	날짜 및 시간
심볼	현재 차트의 심볼 이름을 반환	검사
SymbolInfoDouble	해당 속성에 대한 심볼 double 값을 반환	마켓 정보
SymbolInfoInteger	해당 속성에 지정된 심볼에 대한 정수 유형(long, 날짜시간, int 또는 bool)의 값을 반환	마켓 정보
SymbolInfoMarginRate	주문 유형 및 방향에 따라 마진율을 반환	마켓 정보

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
SymbolInfoSessionQuote	지정된 심볼 및 평일에 대해 지정된 시세 세션의 시작 및 종료 시간을 수신 허용.	마켓 정보
SymbolInfoSessionTrade	지정된 심볼 및 평일에 대해 지정된 트레이딩 세션의 시작 및 종료 시간을 수신 허용.	마켓 정보
SymbolInfoString	해당 속성에 대해 지정된 심볼에 대해 문자열 유형의 값을 반환	마켓 정보
SymbolInfoTick	다음 유형의 변수에 지정된 심볼에 대한 현재 가격을 반환: MqlTick 유형	마켓 정보
SymbolIsSynchronized	터미널에서 선택한 심볼의 데이터가 거래 서버의 데이터와 동기화 되었는지 확인	마켓 정보
SymbolName	지정된 심볼의 이름을 반환	마켓 정보
SymbolSelect	마켓 워치(Market Watch) 창에서 기호를 선택하거나 창에서 심볼을 제거	마켓 정보
SymbolsTotal	사용 가능한 (Market Watch 또는 모두에서 선택된) 심볼 수를 반환	마켓 정보
tan	수의 탄젠트를 반환	산술 함수
TerminalClose	터미널이 작동을 완료하도록 명령	공통 함수
TerminalInfoDouble	mql5 프로그램 환경의 해당 속성의 double 값을 반환	검사
TerminalInfoInteger	mql5 프로그램 환경의 해당 속성의 정수 값을 반환	검사
TerminalInfoString	mql5 프로그램 환경의 해당 속성의 string 값을 반환	검사
TesterStatistics	테스트 결과를 기반으로 계산된 지정된 통계의 값을 반환	공통 함수
TextGetSize	현재의 다음 설정에서 문자열의 너비 및 높이를 반환: 폰트 설정	오브젝트 기능
TextOut	다음을 생성하도록 설계된 사용자 지정 배열(버퍼)로 텍스트를 전송: 그래팩 리소스	오브젝트 기능
TextSetFont	작성 방법(디폴트로 Arial 20 사용)을 사용하여 텍스트를 표시하는 글꼴을 설정	오브젝트 기능

© 2000-2025, MetaQuotes Ltd.

기능	액션	섹션
TimeCurrent	알려진 마지막 서버 시간(마지막 시세 수령 시간)을 날짜시간 형식으로 반환	날짜 및 시간
TimeDaylightSavings	데이라이트 세이빙 타임 스위치 신호를 반환	날짜 및 시간
TimeGMT	클라이언트 터미널이 실행 중인 컴퓨터 로컬 시간별 데이라이트 세이빙 타임으로 GMT 날짜시간 형식으로 반환	날짜 및 시간
TimeGMTOffset	DST 스위치를 고려하여 GMT 시간과 로컬 컴퓨터 시간 사이의 현재 차이(초)를 반환	날짜 및 시간
TimeLocal	로컬 컴퓨터 시간을 날짜시간 형식으로 반환	날짜 및 시간
TimeToString	1970년 1월 1일 이후 경과된 시간(초)을 포함하는 값을 "yyyy.mm.dd hh:mi" 형식의 문자열로 변환	변환 함수
TimeToStruct	날짜시간 값을 MqlDateTime 구조 유형의 변수로 변환	날짜 및 시간
TimeTradeServer	거래 서버의 현재 게산 시간을 반환	날짜 및 시간
UninitializeReason	초기화 취소 사유 코드를 반환	검사
WebRequest	지정된 서버로 HTTP 요청을 전송	공통 함수
ZeroMemory	참조로 전달된 변수를 재설정 변수는 컨스트럭터가 있는 클래스 및 구조를 제외한 모든 유형일 수 있습니다.	공통 함수

© 2000-2025, MetaQuotes Ltd.

MQL5 상수 목록

모든 MQL5 상수 알파벳 순서.

상수	설명	사용법
__DATE__	시간이 없는(시간, 분 및 초가 0인) 파일 컴파일 날짜	인쇄
__DATETIME__	파일 컴파일 날짜 및 시간	인쇄
__FILE__	현재 컴파일된 파일의 이름	인쇄
__FUNCSIG__	매크로가 위치한 본문의 함수 서명. 기능에 대한 전체 설명을 로깅(Logging)하면 오버로드된 기능을 식별하는데	인쇄

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	유용할 수 있습니다	
__FUNCTION__	매크로가 위치한 본문의 함수 이름	인쇄
__LINE__	매크로가 위치한 소스 코드의 라인 번호	인쇄
__MQLBUILD__, __MQL5BUILD__	컴파일러 빌드 번호	인쇄
__PATH__	현재 컴파일 중인 파일의 절대 경로	인쇄
ACCOUNT_ASSETS	계정의 현재 자산	AccountInfoDouble
ACCOUNT_BALANCE	예금 통화 계좌 잔고	AccountInfoDouble
ACCOUNT_COMMISSION_BLOCKED	계정에서 현재 차단된 커	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	미션 금액	
ACCOUNT_COMPANY	계정에 서비스를 제공하는 회사 이름	AccountInfoString
ACCOUNT_CREDIT	예금 통화 계좌 신용	AccountInfoDouble
ACCOUNT_CURRENCY	계좌 통화	AccountInfoString
ACCOUNT_EQUITY	예금 통화 계좌 지분	AccountInfoDouble
ACCOUNT_LEVERAGE	계좌 레버리지	AccountInfoInteger
ACCOUNT_LIABILITIES	계좌의 유동부채	AccountInfoDouble
ACCOUNT_LIMIT_ORDERS	허용되는 최대 활성 보류 중 주문 수	AccountInfoInteger
ACCOUNT_LOGIN	계좌 번호	AccountInfoInteger
ACCOUNT_MARGIN	예금 통화에 사용된 계좌 마진	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ACCOUNT_MARGIN_FREE	예금 통화로 된 계좌의 자유마진	AccountInfoDouble
ACCOUNT_MARGIN_INITIAL	초기 마진. 보류중인 모든 주문의 마진을 충당하기 위해 계정에 예약된 금액	AccountInfoDouble
ACCOUNT_MARGIN_LEVEL	계정 마진 수준 (%)	AccountInfoDouble
ACCOUNT_MARGIN_MAINTENANCE	유지 마진. 모든 미결직의 최소 금액을 충당하기 위해 계정에 예약된 최소 지분	AccountInfoDouble
ACCOUNT_MARGIN_SO_CALL	마진 호출 수준. 설정된 ACCOUNT_MARG	AccountInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	IN_SO_MODE에 따라 백분율 또는 예금 통화로 표시됨	
ACCOUNT_MARGIN_SO_MODE	최소 허용 마진 설정 모드	AccountInfoInteger
ACCOUNT_MARGIN_SO_SO	마진 정지 수준. 설정된 ACCOUNT_MARGIN_SO_MODE에 따라 백분율 또는 예금 통화로 표시됨	AccountInfoDouble
ACCOUNT_NAME	클라이언트 이름	AccountInfoString
ACCOUNT_PROFIT	예금 통화 계좌의 순이익	AccountInfoDouble
ACCOUNT_SERVER	거래 서버 이름	AccountInfoString

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ACCOUNT_STOPOUT_MODE_MONEY	계자 정지 모드 (화폐)	AccountInfoInteger
ACCOUNT_STOPOUT_MODE_PERCENT	계정 정지 모드 (%)	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	현재 계정에 대해 거래 허용됨	AccountInfoInteger
ACCOUNT_TRADE_EXPERT	엑스퍼트 어드바이저에 대해 거래 허용됨	AccountInfoInteger
ACCOUNT_TRADE_MODE	계좌 거래 모드	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	콘테스트 계정	AccountInfoInteger
ACCOUNT_TRADE_MODE_DEMO	데모 계정	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	실제 계정	AccountInfoInteger
ALIGN_CENTER	가운데(편집 개체에 대해서만 해당)	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_LEFT	좌측 정렬	ObjectSetInteger, ObjectGetInteger, ChartScreenShot

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ALIGN_RIGHT	우측 정렬	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	개체 중심에 앵커 포인트 배치	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT	앵커 포인트를 중앙에서 왼쪽으로	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_LOWER	좌측 하단 모서리의 앵커 포인트	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_UPPER	좌측 상단 모서리의 앵커 포인트	ObjectSetInteger, ObjectGetInteger
ANCHOR_LOWER	중앙 아래 앵커 포인트	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT	앵커 포인트를 중앙에서 우측으로	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_LOWER	우측 하단 모서	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	리의 앵커 포인트	
ANCHOR_RIGHT_UPPER	우측 상단 모서리의 앵커 포인트	ObjectSetInteger, ObjectGetInteger
ANCHOR_UPPER	중앙 위 앵커 포인트	ObjectSetInteger, ObjectGetInteger
BASE_LINE	본선	지표선
BOOK_TYPE_BUY	구매 주문 (입찰)	MqlBookInfo
BOOK_TYPE_BUY_MARKET	마켓별 구매 주문	MqlBookInfo
BOOK_TYPE_SELL	판매 주문 (제안)	MqlBookInfo
BOOK_TYPE_SELL_MARKET	마켓별 판매 주문	MqlBookInfo
BORDER_FLAT	납작한 형태	ObjectSetInteger, ObjectGetInteger
BORDER_RAISED	돌출된 형태	ObjectSetInteger, ObjectGetInteger
BORDER_SUNKEN	오목한 형태	ObjectSetInteger, ObjectGetInteger
CHAR_MAX	char 유형으로	숫자 유형 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	나타낼 수 있는 최대 값	
CHAR_MIN	char 유형으로 나타낼 수 있는 최소 값	숫자 유형 상수
CHART_AUTOSCROLL	차트 우측 테두리로 자동 이동 모드	ChartSetInteger, ChartGetInteger
CHART_BARS	일련의 막대로 표시	ChartSetInteger
CHART_BEGIN	차트 시작 (가장 오래된 가격)	ChartNavigate
CHART_BRING_TO_TOP	다른 차트 위에 차트 표시	ChartSetInteger, ChartGetInteger
CHART_CANDLES	일본 캔들스틱 표시	ChartSetInteger
CHART_COLOR_ASK	가격 수준 색상 요청	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
CHART_COLOR_BACKGROUND	차트 배경 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	입찰 가격 레벨 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	베어 캔들스틱의 바디 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BULL	불 캔들스틱의 바디 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_DOWN	베어 캔들스틱 아래 막대, 그림자 및 바디 테두리 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_LINE	선 차트 색상과 "Doji" 일본 캔들스틱의 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_UP	불 캔들스틱의 상부 막대, 그림자 및	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	바디 테두리 색상	
CHART_COLOR_FOREGROUND	축, 스케일 및 OHLC 라인의 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_GRID	격자 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	마지막으로 실행된 거래 가격의 선 색상	ChartSetInteger, ChartGetInteger
CHART_COLOR_STOP_LEVEL	주문 중지 수준의 색상(손실 방지 및 이윤 획득)	ChartSetInteger, ChartGetInteger
CHART_COLOR_VOLUME	볼륨 및 위치 오프닝 레벨 색상	ChartSetInteger, ChartGetInteger
CHART_COMMENT	차트의 코멘트 텍스트	ChartSetString, ChartGetString
CHART_CURRENT_POS	현재 포지션	ChartNavigate

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
CHART_DRAG_TRADE_LEVELS	차트에서 마우스로 트레이딩 레벨을 끌 수 있는 사용 권한. 드래그 모드는 기본으로 활성화 되어 있습니다 (true 값)	ChartSetInteger, ChartGetInteger
CHART_EVENT_MOUSE_MOVE	차트의 모든 mql5 프로그램에 마우스 이동 및 마우스 클릭 이벤트 (CHARTEVENT_MOUSE_MOVE) 알림 전송	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_CREATE	차트의 모	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	든 mql5 프로그램에 새 개체 생성 이벤트 (CHARTEVENT_OBJECT_CREATE) 알림 전송	
CHART_EVENT_OBJECT_DELETE	차트의 모든 mql5 프로그램에 개체 삭제 이벤트 (CHARTEVENT_OBJECT_DELETE) 알림 전송	ChartSetInteger, ChartGetInteger
CHART_FIRST_VISIBLE_BAR	차트에서 첫 번째로 표시되는 막대 수. 막대의 인덱싱은 시계	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	열의 인덱싱과 동일합니다.	
CHART_FIXED_MAX	고정된 차트 최대값	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	고정된 차트 최소값	ChartSetDouble, ChartGetDouble
CHART_FIXED_POSITION	좌측 테두리의 차트 고정 위치 (백분율 값). 차트 고정 위치는 수평 시간 축에 작은 회색 삼각형으로 표시됩니다. 틱이 들어올 때 우측 자동 차트 스크롤이 비활성화된	ChartSetDouble, ChartGetDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	경우에만 표시됩니다 (CHART_AUTOSCROL 속성 참조). 고정 위치의 막대는 확대 및 축소할 때 동일한 위치에 유지됩니다.	
CHART_FOREGROUND	우위의 가격 차트	ChartSetInteger, ChartGetInteger
CHART_HEIGHT_IN_PIXELS	차트 높이 (픽셀)	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	"차트" (OBJ_CHART) 개체를 식별하면 그래픽 개체에 대해 true를 반환. 실제 차트에 대	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	해 false 를 반환	
CHART_LINE	Close 가격에 의해 그려진 선으로 표시	ChartSetInteger
CHART_MODE	차트 유형 (캔들스틱, 막대 또는 선)	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	마우스 좌측 버튼을 사용하여 차트를 수평으로 스크롤. 다음 속성 값이 true로 설정된 경우에도 수직 스크롤을 사용할 수 있습니다: CHART_SC	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	ALEFIX, CHART_SCALEFIX_11 또는 CHART_SCALE_PT_PER_BAR	
CHART_POINTS_PER_BAR	막대 당 포인트 스케일	ChartSetDouble, ChartGetDouble
CHART_PRICE_MAX	차트 최대	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	차트 최소	ChartSetDouble, ChartGetDouble
CHART_SCALE	스케일	ChartSetInteger, ChartGetInteger
CHART_SCALE_PT_PER_BAR	막대 당 포인트로 지정할 스케일	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX	고정 스케일 모드	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX_11	1:1 스케일 모드	ChartSetInteger, ChartGetInteger
CHART_SHIFT	우측 테두리에서 가격 차트 들여쓰	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	기 모드	
CHART_SHIFT_SIZE	우측 테두리의 0 막대 들여쓰기 크기 (%)	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	차트에서 수평으로 요청 (Ask) 값 표시	ChartSetInteger, ChartGetInteger
CHART_SHOW_BID_LINE	차트에서 수평으로 입찰 (BID) 값 표시	ChartSetInteger, ChartGetInteger
CHART_SHOW_DATE_SCALE	차트에 시간 스케일 표시	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	차트에 격자 표시	ChartSetInteger, ChartGetInteger
CHART_SHOW_LAST_LINE	차트에 수평으로 마지막 (Last) 값 표시	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
CHART_SHOW_OBJECT_DESCR	그래픽 개체에 대한 팝업 설명	ChartSetInteger, ChartGetInteger
CHART_SHOW_OHLC	좌측 상단 모서리에 OHLC 값 표시	ChartSetInteger, ChartGetInteger
CHART_SHOW_ONE_CLICK	표시: "원클릭 트레이딩"차트의 패널	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	인접 기간 사이에 수직 분리자 표시	ChartSetInteger, ChartGetInteger
CHART_SHOW_PRICE_SCALE	차트에 가격 스케일을 표시	ChartSetInteger, ChartGetInteger
CHART_SHOW_TRADE_LEVELS	차트에 거래 수준 표시 (오픈 포지션 수준, 손실 중지, 이익 획득 및 보	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	류 중인 주문)	
CHART_SHOW_VOLUMES	차트에 폴륨 표시	ChartSetInteger, ChartGetInteger
CHART_VISIBLE_BARS	차트에 표시할 수 있는 막대의 수	ChartSetInteger, ChartGetInteger
CHART_VOLUME_HIDE	벌륨이 표시되지 않음	ChartSetInteger
CHART_VOLUME_REAL	거래 볼륨	ChartSetInteger
CHART_VOLUME_TICK	틱 볼륨	ChartSetInteger
CHART_WIDTH_IN_BARS	차트 너비 (막대)	ChartSetInteger, ChartGetInteger
CHART_WIDTH_IN_PIXELS	차트 너비 (픽셀)	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	차트 창 핸들 (HWND)	ChartSetInteger, ChartGetInteger
CHART_WINDOW_IS_VISIBLE	하위 창 가시성	ChartSetInteger, ChartGetInteger
CHART_WINDOW_YDISTANCE	수직으로 Y 축을 따라 지표	ChartSetInteger, ChartGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	하위 창의 상단 프레임과 기본 차트 창의 프레임 사이의 거리 (픽셀) 마우스 이벤트의 경우 커서 좌표는 기본 차트 창의 좌표를 기준으로 전달되며, 지표 하위 창의 그래픽 개체 좌표는 하위 창의 좌측 상단 모서리를 기준으로 설정됩니다. 이 값은 기본 차	

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	트의 절대 좌표를 하위 창의 로컬 좌표로 변환하여 하위 창 프레임의 좌측 상단 모서리를 기준으로 설정된 그래픽 개체와 올바르게 작업하는 데 필요합니다.	
CHART_WINDOWS_TOTAL	지표 하위 창을 포함한 차트 창의 총 수	ChartSetInteger, ChartGetInteger
CHARTEVENT_CHART_CHANGE	속성 대화 상자를 통한 차트 크기 변경 또	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	는 차트 속성 수정	
CHARTEVENT_CLICK	차트 클릭	OnChartEvent
CHARTEVENT_CUSTOM	사용자 지정 이벤트 범위의 초기 이벤트 수	OnChartEvent
CHARTEVENT_CUSTOM_LAST	사용자 지정 이벤트 범위에서 이벤트의 최종 번호	OnChartEvent
CHARTEVENT_KEYDOWN	키 입력	OnChartEvent
CHARTEVENT_MOUSE_MOVE	마우스 이동, 마우스 클릭 (CHART_EVENT_MOUSE_MOVE=true가 설정된 경우)	OnChartEvent
CHARTEVENT_OBJECT_CHANGE	그래픽 개체 속	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	성이 속성 대화 상자를 통해 변경됨	
CHARTEVENT_OBJECT_CLICK	클릭: 그래픽 개체	OnChartEvent
CHARTEVENT_OBJECT_CREATE	그래픽 개체가 생성됨 (CHART_EVENT_OBJECT_CREATE=true 가 설정된 경우)	OnChartEvent
CHARTEVENT_OBJECT_DELETE	그래픽 개체가 삭제됨 (CHART_EVENT_OBJECT_DELETE=true 가 설정된 경우)	OnChartEvent
CHARTEVENT_OBJECT_DRAG	그래픽 개체 끌어서 놓기	OnChartEvent

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
CHARTEVENT_OBJECT_ENDEDIT	그래픽 개체 편집에서 텍스트 끝 편집	OnChartEvent
CHARTS_MAX	터미널에서 동시에 열려 있는 최대 차트 수	기타 상수
CHIKOUSPAN_LINE	지구(Chikou) 스팬 선	지표선
clrAliceBlue	앨리스 블루	웹 색상
clrAntiqueWhite	앤틱 화이트	웹 색상
clrAqua	아쿠아	웹 색상
clrAquamarine	아쿠아마린	웹 색상
clrBeige	베이지	웹 색상
clrBisque	비스크	웹 색상
clrBlack	블랙	웹 색상
clrBlanchedAlmond	밝은 아몬드	웹 색상
clrBlue	블루	웹 색상

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
clrBlueViolet	블루 바이올렛	웹 색상
clrBrown	브라운	웹 색상
clrBurlyWood	밝은 브라운	웹 색상
clrCadetBlue	카데트 블루	웹 색상
clrChartreuse	샤르트뢰즈	웹 색상
clrChocolate	초콜릿	웹 색상
clrCoral	코럴	웹 색상
clrCornflowerBlue	콘플라워 블루	웹 색상
clrCornsilk	콘실크	웹 색상
clrCrimson	크림슨	웹 색상
clrDarkBlue	어두운 블루	웹 색상
clrDarkGoldenrod	어두운 진황색	웹 색상
clrDarkGray	어두운 그레이	웹 색상
clrDarkGreen	어두운 그린	웹 색상
clrDarkKhaki	어두운 카키	웹 색상

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
clrDarkOliveGreen	어두운 올리브 그린	웹 색상
clrDarkOrange	어두운 오렌지	웹 색상
clrDarkOrchid	어두운 오키드	웹 색상
clrDarkSalmon	어두운 연어색	웹 색상
clrDarkSeaGreen	어두운 바다 녹색	웹 색상
clrDarkSlateBlue	어두운 슬레이트 블루	웹 색상
clrDarkSlateGray	어두운 슬레이트 그레이	웹 색상
clrDarkTurquoise	어두운 터키석색	웹 색상
clrDarkViolet	어두운 바이올렛	웹 색상
clrDeepPink	진한 분홍	웹 색상
clrDeepSkyBlue	진한 하늘색	웹 색상
clrDimGray	진한 회색	웹 색상

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
clrDodgerBlue	다저 블루	웹 색상
clrFireBrick	파이어 브릭	웹 색상
clrForestGreen	포레스트 그린	웹 색상
clrGainsboro	게인즈버러	웹 색상
clrGold	금색	웹 색상
clrGoldenrod	진황색	웹 색상
clrGray	회색	웹 색상
clrGreen	녹색	웹 색상
clrGreenYellow	그린 옐로우	웹 색상
clrHoneydew	허니듀	웹 색상
clrHotPink	핫 핑크	웹 색상
clrIndianRed	인디언 레드	웹 색상
clrIndigo	인디고	웹 색상
clrIvory	상아색	웹 색상
clrKhaki	카키	웹 색상
clrLavender	라벤더	웹 색상
clrLavenderBlush	라벤더 블러쉬	웹 색상
clrLawnGreen	잔디색	웹 색상

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
clrLemonChiffon	레몬 시폰	웹 색상
clrLightBlue	밝은 파랑	웹 색상
clrLightCoral	밝은 코럴	웹 색상
clrLightCyan	밝은 청녹색	웹 색상
clrLightGoldenrod	밝은 진황색	웹 색상
clrLightGray	밝은 회색	웹 색상
clrLightGreen	밝은 녹색	웹 색상
clrLightPink	연분홍	웹 색상
clrLightSalmon	밝은 연어색	웹 색상
clrLightSeaGreen	밝은 바다 녹색	웹 색상
clrLightSkyBlue	밝은 하늘색	웹 색상
clrLightSlateGray	밝은 슬레이트 그레이	웹 색상
clrLightSteelBlue	밝은 스틸 블루	웹 색상
clrLightYellow	밝은 노랑색	웹 색상
clrLime	라임	웹 색상

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
clrLimeGreen	라임 그린	웹 색상
clrLinen	리넨	웹 색상
clrMagenta	마젠타	웹 색상
clrMaroon	밤색	웹 색상
clrMediumAquamarine	미디움 아쿠아마린	웹 색상
clrMediumBlue	미디움 파란색	웹 색상
clrMediumOrchid	미디움 오키드	웹 색상
clrMediumPurple	미디움 퍼플	웹 색상
clrMediumSeaGreen	미디움 바다 녹색	웹 색상
clrMediumSlateBlue	미디움 슬레이트 블루	웹 색상
clrMediumSpringGreen	미디움 스프링 그린	웹 색상
clrMediumTurquoise	미디움 터키석색	웹 색상
clrMediumVioletRed	미디움 바이올렛 레드	웹 색상

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
clrMidnightBlue	미드나잇 블루	웹 색상
clrMintCream	민트 크림	웹 색상
clrMistyRose	미스티 로즈	웹 색상
clrMoccasin	모카신	웹 색상
clrNavajoWhite	나바호 화이트	웹 색상
clrNavy	네이비	웹 색상
clrNONE	무색	기타 상수
clrOldLace	올드 레이스	웹 색상
clrOlive	올리브	웹 색상
clrOliveDrab	올리브 드랩	웹 색상
clrOrange	주황색	웹 색상
clrOrangeRed	적황색	웹 색상
clrOrchid	오키드	웹 색상
clrPaleGoldenrod	흐린 진황색	웹 색상
clrPaleGreen	흐린 녹색	웹 색상
clrPaleTurquoise	흐린 터키석색	웹 색상

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
clrPaleVioletRed	흐린 바이올렛 레드	웹 색상
clrPapayaWhip	파파야 윕	웹 색상
clrPeachPuff	피치 퍼프	웹 색상
clrPeru	페루	웹 색상
clrPink	분홍색	웹 색상
clrPlum	플럼	웹 색상
clrPowderBlue	파우더 블루	웹 색상
clrPurple	퍼플	웹 색상
clrRed	빨강색	웹 색상
clrRosyBrown	로지 브라운	웹 색상
clrRoyalBlue	로얄 블루	웹 색상
clrSaddleBrown	새들 브라운	웹 색상
clrSalmon	연어색	웹 색상
clrSandyBrown	샌디 브라운	웹 색상
clrSeaGreen	바다 녹색	웹 색상
clrSeashell	바다 조개색	웹 색상
clrSienna	시에나	웹 색상

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
clrSilver	은색	웹 색상
clrSkyBlue	하늘색	웹 색상
clrSlateBlue	슬레이트 블루	웹 색상
clrSlateGray	슬레이트 그레이	웹 색상
clrSnow	스노우	웹 색상
clrSpringGreen	스프링 그린	웹 색상
clrSteelBlue	스틸 블루	웹 색상
clrTan	탠	웹 색상
clrTeal	청록색	웹 색상
clrThistle	시슬	웹 색상
clrTomato	토마토	웹 색상
clrTurquoise	터키석색	웹 색상
clrViolet	바이올렛	웹 색상
clrWheat	오트	웹 색상
clrWhite	흰색	웹 색상
clrWhiteSmoke	화이트 스모크	웹 색상
clrYellow	노랑색	웹 색상
clrYellowGreen	연두색	웹 색상
CORNER_LEFT_LOWER	좌표 중심	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	이 차트 좌측 하단에 있습니다	
CORNER_LEFT_UPPER	좌표 중심이 차트 좌측 상단에 있습니다	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_LOWER	좌표 중심이 차트 우측 하단에 있습니다	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_UPPER	좌표 중심이 차트 우측 상단에 있습니다	ObjectSetInteger, ObjectGetInteger
CP_ACP	현재 윈도우 (Windows) ANSI 코드 페이지.	CharArrayToString, StringToCharArray, FileOpen
CP_MACCP	현재 시스템 맥 (Macintosh) 코드	CharArrayToString, StringToCharArray, FileOpen

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	페이지. 참고: 이 값은 이전에 생성된 프로그램 코드에서 주로 사용되며, 오늘날의 맥(Macintosh) 컴퓨터에서는 인코딩에 유니코드를 사용하기 때문에 지금은 사용할 수 없습니다.	
CP_OEMCP	현재 시스템 OEM 코드 페이지.	CharArrayToString, StringToCharArray, FileOpen
CP_SYMBOL	심볼 코드 페이지	CharArrayToString, StringToCharArray, FileOpen

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
CP_THREAD_ACP	현재 스레드에 대한 Windows ANSI 코드 페이지.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF7	UTF-7 코드 페이지.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF8	UTF-8 코드 페이지.	CharArrayToString, StringToCharArray, FileOpen
CRYPT_AES128	128 비트 키(16 바이트)를 사용한 AES 암호화	CryptEncode, CryptDecode
CRYPT_AES256	256 비트 키(32 바이트)를 사용한 AES 암호화	CryptEncode, CryptDecode
CRYPT_ARCH_ZIP	ZIP 아카이브	CryptEncode, CryptDecode
CRYPT_BASE64	BASE64	CryptEncode, CryptDecode

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
CRYPT_DES	56 비트 키(7 바이트)를 사용한 DES 암호화	CryptEncode, CryptDecode
CRYPT_HASH_MD5	MD5 HASH 계산	CryptEncode, CryptDecode
CRYPT_HASH_SHA1	SHA1 HASH 계산	CryptEncode, CryptDecode
CRYPT_HASH_SHA256	SHA256 HASH 계산	CryptEncode, CryptDecode
DBL_DIG	double 형식의 유효 소수 자릿수	숫자 유형 상수
DBL_EPSILON	조건을 충족하는 최소값: 1.0+DBL_EPSILON != 1.0 (for double type)	숫자 유형 상수
DBL_MANT_DIG	double 유형에 대한 비트 수(가수)	숫자 유형 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
DBL_MAX	double 유형으로 나타낼 수 있는 최대 값	숫자 유형 상수
DBL_MAX_10_EXP	double 유형에 대한 지수 등급의 최대 십진수 값	숫자 유형 상수
DBL_MAX_EXP	double 유형에 대한 지수 등급의 최대 이진수 값	숫자 유형 상수
DBL_MIN	double 유형으로 나타낼 수 있는 최소 값(양수)	숫자 유형 상수
DBL_MIN_10_EXP	double 유형에 대한 지수 등급의 최소 십진수 값	숫자 유형 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
DBL_MIN_EXP	double 유형에 대한 지수 등급의 최소 이진수 값	숫자 유형 상수
DEAL_COMMENT	거래 코멘트	HistoryDealGetString
DEAL_COMMISSION	거래 수수료	HistoryDealGetDouble
DEAL_ENTRY	거래 입력 – 엔트리 인, 엔트리 아웃, 반전	HistoryDealGetInteger
DEAL_ENTRY_IN	엔트리 인	HistoryDealGetInteger
DEAL_ENTRY_INOUT	반전	HistoryDealGetInteger
DEAL_ENTRY_OUT	엔트리 아웃	HistoryDealGetInteger
DEAL_MAGIC	거래 매직 넘버 (ORDER_MAGIC 참조)	HistoryDealGetInteger
DEAL_ORDER	거래 주문 번호	HistoryDealGetInteger
DEAL_POSITION_ID	포지션 식별자: 이 거	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	래가 참여한 시작, 수정 또는 변경. 각 포지션에는 위치의 전체 스팬 동안 심볼에 대해 실행된 모든 거래에 할당되는 고유의 식별자가 있습니다.	
DEAL_PRICE	거래 가격	HistoryDealGetDouble
DEAL_PROFIT	거래 이익	HistoryDealGetDouble
DEAL_SWAP	마감 시 누적 스왑	HistoryDealGetDouble
DEAL_SYMBOL	거래 심볼	HistoryDealGetString
DEAL_TIME	거래 시간	HistoryDealGetInteger
DEAL_TIME_MSC	1970년 1월 1일 이후 거래 실	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	행 시간(밀리초)	
DEAL_TYPE	거래 유형	HistoryDealGetInteger
DEAL_TYPE_BALANCE	잔고	HistoryDealGetInteger
DEAL_TYPE_BONUS	보너스	HistoryDealGetInteger
DEAL_TYPE_BUY	구매	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	취소된 구매 거래. 이전에 실행된 구매 거래가 취소되는 상황이 발생할 수 있습니다. 이 경우, 이전에 실행된 거래 (DEAL_TYPE_BUY)의 유형이 DEAL_TYPE_BUY_CANCELED로 변경되고, 손익/손실이 0으	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	로 설정됩니다. 이전에 취득한 손익은 분리된 잔액 연산을 사용하여 과금/출금됩니다	
DEAL_TYPE_CHARGE	추가 과금	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	추가 수수료	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	일일 에이전트 수수료	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	월 에이전트 수수료	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_DAILY	일일 수수료	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_MONTHLY	월별 수수료	HistoryDealGetInteger
DEAL_TYPE_CORRECTION	보정	HistoryDealGetInteger
DEAL_TYPE_CREDIT	신용	HistoryDealGetInteger
DEAL_TYPE_INTEREST	이자율	HistoryDealGetInteger
DEAL_TYPE_SELL	판매	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
DEAL_TYPE_SELL_CANCELED	취소된 판매 거래. 이전에 실행된 판매 거래가 취소되는 상황이 발생할 수 있습니다. 이 경우, 이전에 실행된 거래 (DEAL_TYPE_SELL)의 유형이 DEAL_TYPE_SELL_CANCELED로 변경되고, 손익/손실이 0으로 설정됩니다. 이전에 취득한 손익은 분리된 잔액 연산을	HistoryDealGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	사용하여 과금/출금 됩니다	
DEAL_VOLUME	거래 볼륨	HistoryDealGetDouble
DRAW_ARROW	화살표 그리기	그리기 스타일
DRAW_BARS	일련의 막대로 표시	그리기 스타일
DRAW_CANDLES	일련의 캔들스틱으로 표시	그리기 스타일
DRAW_COLOR_ARROW	멀티 컬러의 화살표 그리기	그리기 스타일
DRAW_COLOR_BARS	멀티 컬러 막대	그리기 스타일
DRAW_COLOR_CANDLES	멀티 컬러 캔들스틱	그리기 스타일
DRAW_COLOR_HISTOGRAM	0 라인의 멀티 히스토그램	그리기 스타일
DRAW_COLOR_HISTOGRAM2	두 지표 버퍼의 멀티	그리기 스타일

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	컬러 히스토그램	
DRAW_COLOR_LINE	멀티 컬러 선	그리기 스타일
DRAW_COLOR_SECTION	멀티 컬러 섹션	그리기 스타일
DRAW_COLOR_ZIGZAG	멀티 컬러 지그재그 (ZigZag)	그리기 스타일
DRAW_FILLING	두 수준 사이의 색상 채우기	그리기 스타일
DRAW_HISTOGRAM	0라인의 히스토그램	그리기 스타일
DRAW_HISTOGRAM2	두 지표 버퍼의 히스토그램	그리기 스타일
DRAW_LINE	선	그리기 스타일
DRAW_NONE	그려지지 않음	그리기 스타일
DRAW_SECTION	섹션	그리기 스타일
DRAW_ZIGZAG	스타일 지그재그 (Zigza	그리기 스타일

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	g)로 막대에서 수직 섹션 허용	
ELLIOTT_CYCLE	싸이클	ObjectSetInteger, ObjectGetInteger
ELLIOTT_GRAND_SUPERCYCLE	그랜드 수퍼싸이클 (Grand Supercycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_INTERMEDIATE	중급	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINOR	마이너	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUETTE	미뉴에트	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUTE	분	ObjectSetInteger, ObjectGetInteger
ELLIOTT_PRIMARY	초급	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUBMINUETTE	하위 미뉴에트	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUPERCYCLE	수퍼싸이클	ObjectSetInteger, ObjectGetInteger
EMPTY_VALUE	지표 버퍼의 빈 값	기타 상수
ERR_ACCOUNT_WRONG_PROPERTY	잘못된 계정 속성 ID	GetLastError
ERR_ARRAY_BAD_SIZE	요청된 배열 크기가 2	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	GB를 초과	
ERR_ARRAY_RESIZE_ERROR	메모리가 부족하여 배열을 재배치할 수 없거나 정적 배열의 크기를 변경할 수 없습니다	GetLastError
ERR_BOOKS_CANNOT_ADD	Depth Of Market를 추가할 수 없습니다	GetLastError
ERR_BOOKS_CANNOT_DELETE	Depth Of Market를 제거할 수 없습니다	GetLastError
ERR_BOOKS_CANNOT_GET	Depth Of Market에서 데이터를 가져올 수 없습니다	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_BOOKS_CANNOT_SUBSCRIBE	Depth Of Market에서 새 데이터를 수신하기 위해 구독 중 오류 발생	GetLastError
ERR_BUFFERS_NO_MEMORY	메모리가 부족하여 지표 버퍼를 배포할 수 없습니다	GetLastError
ERR_BUFFERS_WRONG_INDEX	잘못된 지표 버퍼 인덱스	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	디렉토리를 지우지 못했습니다(하나 이상의 파일이 차단되었을 수 있어 제거 작업에 실패	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	했습니다)	
ERR_CANNOT_DELETE_DIRECTORY	디렉토리를 제거할 수 없습니다	GetLastError
ERR_CANNOT_DELETE_FILE	파일 삭제 오류	GetLastError
ERR_CANNOT_OPEN_FILE	파일 열기 오류	GetLastError
ERR_CHAR_ARRAY_ONLY	char 유형의 배열이어야 합니다	GetLastError
ERR_CHART_CANNOT_CHANGE	차트 심볼 및 기간을 변경하는 데 실패하였습니다	GetLastError
ERR_CHART_CANNOT_CREATE_TIMER	타이머 생성에 실패하였습니다	GetLastError
ERR_CHART_CANNOT_OPEN	차트 열기 오류	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	차트에 지	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	표 추가 중 오류 발생	
ERR_CHART_INDICATOR_CANNOT_DEL	차트에서 지표 삭제 중 오류 발생	GetLastError
ERR_CHART_INDICATOR_NOT_FOUND	지정된 차트에서 지시자를 찾을 수 없음	GetLastError
ERR_CHART_NAVIGATE_FAILED	차트 탐색 중 오류 발생	GetLastError
ERR_CHART_NO_EXPERT	차트에는 이벤트를 처리할 수 있는 엑스퍼트 어드바이저가 없음	GetLastError
ERR_CHART_NO_REPLY	차트가 응답하지 않음	GetLastError
ERR_CHART_NOT_FOUND	차트를 찾을 수	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	없습니다	
ERR_CHART_SCREENSHOT_FAILED	스크린샷 생성 중 오류	GetLastError
ERR_CHART_TEMPLATE_FAILED	템플릿 적용 중 오류 발생	GetLastError
ERR_CHART_WINDOW_NOT_FOUND	지표가 포함된 하위 창을 찾을 수 없음	GetLastError
ERR_CHART_WRONG_ID	잘못된 차트 ID	GetLastError
ERR_CHART_WRONG_PARAMETER	차트 작업 함수에 대한 매개변수의 오류 값	GetLastError
ERR_CHART_WRONG_PROPERTY	잘못된 차트 속성 ID	GetLastError
ERR_CUSTOM_WRONG_PROPERTY	잘못된 사용자 지정 지표 속성의 ID	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_DIRECTORY_NOT_EXIST	디렉토리가 존재하지 않음	GetLastError
ERR_DOUBLE_ARRAY_ONLY	double 유형의 배열이어야 합니다	GetLastError
ERR_FILE_BINSTRINGSIZE	파일이 바이너리 파일로 열렸으므로 문자열 크기를 지정해야 합니다	GetLastError
ERR_FILE_CACHEBUFFER_ERROR	캐시를 읽을 수 있는 메모리가 부족함	GetLastError
ERR_FILE_CANNOT_REWRITE	파일을 다시 쓸 수 없습니다	GetLastError
ERR_FILE_IS_DIRECTORY	파일이 아니라 디렉토리	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	입니다	
ERR_FILE_ISNOT_DIRECTORY	디렉토리가 아니라 파일입니다	GetLastError
ERR_FILE_NOT_EXIST	파일이 존재하지 않습니다	GetLastError
ERR_FILE_NOTBIN	파일을 바이너리 파일로 열어야 합니다	GetLastError
ERR_FILE_NOTCSV	파일을 CSV로 열어야 합니다	GetLastError
ERR_FILE_NOTTOREAD	읽기를 하려면 파일을 열어야 합니다	GetLastError
ERR_FILE_NOTTOWRITE	쓰기를 하려면 파일을 열어야 합니다	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_FILE_NOTTXT	파일을 텍스트로 열어야 합니다	GetLastError
ERR_FILE_NOTTXTORCSV	파일을 텍스트 또는 CSV로 열어야 합니다	GetLastError
ERR_FILE_READERROR	파일 읽기 오류	GetLastError
ERR_FILE_WRITEERROR	파일에 리소스 쓰기 실패	GetLastError
ERR_FLOAT_ARRAY_ONLY	float 유형의 배열이어야 합니다	GetLastError
ERR_FTP_SEND_FAILED	ftp를 통한 파일 전송 실패	GetLastError
ERR_FUNCTION_NOT_ALLOWED	기능이 호출에 허용되지 않습니다	GetLastError
ERR_GLOBALVARIABLE_EXISTS	동일한 이	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	름을 가진 클라이언트 터미널의 글로벌 변수가 이미 있습니다	
ERR_GLOBALVARIABLE_NOT_FOUND	클라이언트 터미널의 글로벌 변수를 찾을 수 없습니다	GetLastError
ERR_HISTORY_NOT_FOUND	요청된 기록을 찾을 수 없습니다	GetLastError
ERR_HISTORY_WRONG_PROPERTY	잘못된 내역 속성의 ID	GetLastError
ERR_INCOMPATIBLE_ARRAYS	호환되지 않는 배열 복사. 문자열 배열은 문자열 배	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	열에만 복사할 수 있으며 숫자 배열에는 숫자 배열만 복사할 수 있습니다	
ERR_INCOMPATIBLE_FILE	텍스트 파일은 문자열 배열용이어야 하며 다른 배열용은 바이너리입니다	GetLastError
ERR_INDICATOR_CANNOT_ADD	차트에 지표 적용 중 오류 발생	GetLastError
ERR_INDICATOR_CANNOT_APPLY	지표를 다른 지표에 적용할 수 없습니다	GetLastError
ERR_INDICATOR_CANNOT_CREATE	지표를 생	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	성할 수 없습니다	
ERR_INDICATOR_CUSTOM_NAME	배열의 첫 번째 매개변수는 사용자 지정 지표의 이름이어야 합니다	GetLastError
ERR_INDICATOR_DATA_NOT_FOUND	요청된 데이터를 찾을 수 없음	GetLastError
ERR_INDICATOR_NO_MEMORY	메모리가 부족하여 지표를 추가할 수 없습니다	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	지표를 생성할 때 배열에 잘못된 매개변수 유형이 있습니다	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_INDICATOR_PARAMETERS_MISSING	지표 생성 시 매개변수 없음	GetLastError
ERR_INDICATOR_UNKNOWN_SYMBOL	알려지지 않은 심볼	GetLastError
ERR_INDICATOR_WRONG_HANDLE	잘못된 지표 핸들	GetLastError
ERR_INDICATOR_WRONG_INDEX	잘못된 요청 지표 버퍼의 인덱스	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	지표 생성 시 잘못된 매개변수 수	GetLastError
ERR_INT_ARRAY_ONLY	int 유형의 배열이어야 합니다	GetLastError
ERR_INTERNAL_ERROR	예상치 못한 내부 오류	GetLastError
ERR_INVALID_ARRAY	잘못된 유형, 잘못된 크기	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	또는 동적 배열의 손상된 개체의 배열	
ERR_INVALID_DATETIME	유효하지 않은 날짜 또는 시간	GetLastError
ERR_INVALID_FILEHANDLE	이 핸들이 있는 파일이 닫혔거나 전혀 열리지 않습니다	GetLastError
ERR_INVALID_PARAMETER	시스템 기능을 호출 시 잘못된 매개변수	GetLastError
ERR_INVALID_POINTER	잘못된 포인터	GetLastError
ERR_INVALID_POINTER_TYPE	잘못된 포인터 유형	GetLastError
ERR_LONG_ARRAY_ONLY	long 유형의 배열이어야	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	합니다	
ERR_MAIL_SEND_FAILED	이메일 전송 실패	GetLastError
ERR_MARKET_LASTTIME_UNKNOWN	마지막 틱 시간을 알 수 없음(틱 없음)	GetLastError
ERR_MARKET_NOT_SELECTED	MarketWatch에서 심볼이 선택되지 않음	GetLastError
ERR_MARKET_SELECT_ERROR	MarketWatch에서 심볼 추가 또는 삭제 오류	GetLastError
ERR_MARKET_UNKNOWN_SYMBOL	알려지지 않은 심볼	GetLastError
ERR_MARKET_WRONG_PROPERTY	잘못된 심볼 속성의 식별자	GetLastError
ERR_MQL5_WRONG_PROPERTY	잘못된 프로그램 속성의	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	식별자	
ERR_NO_STRING_DATE	문자열에 날짜 없음	GetLastError
ERR_NOT_ENOUGH_MEMORY	메모리가 부족하여 시스템 기능을 수행할 수 없습니다	GetLastError
ERR_NOTIFICATION_SEND_FAILED	다음을 전송 실패: 알림	GetLastError
ERR_NOTIFICATION_TOO_FREQUENT	알림 전송이 너무 잦음	GetLastError
ERR_NOTIFICATION_WRONG_PARAMETER	알림 전송에 잘못된 매개변수입니다. 빈 문자열 또는 NULL이 SendNotification() 함수로 전달되	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	었습니다	
ERR_NOTIFICATION_WRONG_SETTINGS	잘못된 터미널 알림 설정 (ID가 지정되지 않았거나 사용 권한이 설정되지 않음)	GetLastError
ERR_NOTINITIALIZED_STRING	초기화되지 않은 문자열	GetLastError
ERR_NUMBER_ARRAYS_ONLY	숫자 배열이어야 합니다	GetLastError
ERR_OBJECT_ERROR	그래픽 개체 작업 중 오류	GetLastError
ERR_OBJECT_GETDATE_FAILED	값에 해당하는 날짜를 가져올 수 없습니다	GetLastError
ERR_OBJECT_GETVALUE_FAILED	날짜에 해당하	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	는 값을 가져올 수 없습니다	
ERR_OBJECT_NOT_FOUND	그래픽 개체를 찾을 수 없습니다	GetLastError
ERR_OBJECT_WRONG_PROPERTY	잘못된 그래픽 개체 속성 ID	GetLastError
ERR_ONEDIM_ARRAYS_ONLY	1차원 배열이어야 합니다	GetLastError
ERR_OPENCL_BUFFER_CREATE	다음을 생성 실패: OpenCL 버퍼	GetLastError
ERR_OPENCL_CONTEXT_CREATE	다음을 생성 실패: OpenCL 컨텍스트	GetLastError
ERR_OPENCL_EXECUTE	OpenCL 프로그램 런타임 오류	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_OPENCL_INTERNAL	다음 중 내부 오류 발생: running OpenCL 실행	GetLastError
ERR_OPENCL_INVALID_HANDLE	유효하지 않은 OpenCL handle	GetLastError
ERR_OPENCL_KERNEL_CREATE	다음을 생성하는 데 오류: OpenCL 커널	GetLastError
ERR_OPENCL_NOT_SUPPORTED	OpenCL 기능은 이 컴퓨터에서 지원되지 않습니다	GetLastError
ERR_OPENCL_PROGRAM_CREATE	오류 발생: OpenCL 프로그램 컴파일하는 도중	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_OPENCL_QUEUE_CREATE	OpenCL에서 실행 대기열을 생성하는데 실패	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	오류 발생: 매개변수 설정 (OpenCL 커널)	GetLastError
ERR_OPENCL_TOO_LONG_KERNEL_NAME	너무 긴 커널 이름 (OpenCL 커널)	GetLastError
ERR_OPENCL_WRONG_BUFFER_OFFSET	OpenCL 버퍼의 유효하지 않은 오프셋	GetLastError
ERR_OPENCL_WRONG_BUFFER_SIZE	유효하지 않은 OpenCL 버퍼 크기	GetLastError
ERR_PLAY_SOUND_FAILED	사운드 재생 실패	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	일치하는	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	동적 및 정적 리소스 이름	
ERR_RESOURCE_NAME_IS_TOO_LONG	리소스 이름이 63자를 초과합니다	GetLastError
ERR_RESOURCE_NOT_FOUND	EX5에서 이 이름의 리소스를 찾을 수 없습니다	GetLastError
ERR_RESOURCE_UNSUPPORTED_TYPE	지원되지 않는 리소스 유형이거나 또는 크기가 16 Mb를 초과합니다	GetLastError
ERR_SERIES_ARRAY	시계열을 사용할 수 없습니다	GetLastError
ERR_SHORT_ARRAY_ONLY	짧은 유형의 배열이어야	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	합니다	
ERR_SMALL_ARRAY	배열이 너무 작아서 시작 포지션이 배열 밖에 있음	GetLastError
ERR_SMALL_ASSERIES_ARRAY	수신 배열이 AS_SERIES로 선언되었으며 크기가 부족합니다	GetLastError
ERR_STRING_OUT_OF_MEMORY	문자열에 사용할 메모리가 부족합니다	GetLastError
ERR_STRING_RESIZE_ERROR	메모리가 부족하여 문자열을 재배치할 수 없습니다	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_STRING_SMALL_LEN	문자열 길이가 예상보다 짧습니다	GetLastError
ERR_STRING_TIME_ERROR	문자열을 현재로 변환하는 중 오류 발생	GetLastError
ERR_STRING_TOO_BIGNUMBER	너무 크며 ULONG_MAX보다 큰 수	GetLastError
ERR_STRING_UNKNOWNTYPE	문자열로 변환할 대 알 수 없는 데이터 유형	GetLastError
ERR_STRING_ZEROADDED	문자열 끝에 0이 추가되어 사용할 수 없는 작업	GetLastError
ERR_STRINGPOS_OUTOFRANGE	문자열 외부의 포지션	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_STRUCT_WITHOBJECTS_ORCLASS	구조에는 문자열 개체 또는 동적 배열 또는 이러한 개체 또는 클래스의 구조가 포함됩니다	GetLastError
ERR_SUCCESS	작업이 성공적으로 완료되었습니다	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	잘못된 터미널 속성 식별자	GetLastError
ERR_TOO_LONG_FILENAME	너무 긴 파일 이름	GetLastError
ERR_TOO_MANY_FILES	동시에 64개 이상의 파일을 열 수 없습니다	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_TOO_MANY_FORMATTERS	매개변수보다 많은 형식 지정자의 양	GetLastError
ERR_TOO_MANY_PARAMETERS	형식 지정자보다 많은 매개변수의 양	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	거래를 찾을 수 없습니다	GetLastError
ERR_TRADE_DISABLED	엑스퍼트 어드바이저 트레이딩 금지	GetLastError
ERR_TRADE_ORDER_NOT_FOUND	주문을 찾을 수 없습니다	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	포지션을 찾을 수 없습니다	GetLastError
ERR_TRADE_SEND_FAILED	거래 요청 전송 실패	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ERR_TRADE_WRONG_PROPERTY	잘못된 거래 속성 ID	GetLastError
ERR_USER_ERROR_FIRST	사용자 정의 오류는 이 코드로 시작합니다	GetLastError
ERR_WEBREQUEST_CONNECT_FAILED	지정된 URL에 연결 실패	GetLastError
ERR_WEBREQUEST_INVALID_ADDRESS	유효하지 않은 URL	GetLastError
ERR_WEBREQUEST_REQUEST_FAILED	HTTP 요청 실패	GetLastError
ERR_WEBREQUEST_TIMEOUT	시간 초과	GetLastError
ERR_WRONG_DIRECTORYNAME	잘못된 디렉토리 이름	GetLastError
ERR_WRONG_FILEHANDLE	잘못된 파일 핸들	GetLastError
ERR_WRONG_FILENAME	잘못된 파일 이름	GetLastError
ERR_WRONG_FORMATSTRING	잘못된	GetLastError

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	string 서식	
ERR_WRONG_INTERNAL_PARAMETER	클라이언트 터미널 기능 내부 호출의 잘못된 매개변수	GetLastError
ERR_WRONG_STRING_DATE	잘못된 날짜 (string)	GetLastError
ERR_WRONG_STRING_OBJECT	손상된 string 개체	GetLastError
ERR_WRONG_STRING_PARAMETER	손상된 string 유형 매개변수	GetLastError
ERR_WRONG_STRING_TIME	잘못된 시간 (string)	GetLastError
ERR_ZEROSIZE_ARRAY	0 길이의 배열	GetLastError
FILE_ACCESS_DATE	파일에 마지막으로 액세스한 날짜	FileGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
FILE_ANSI	ANSI 유형 String (1 바이트 심볼). FileOpen()에서 플래그 사용.	FileOpen
FILE_BIN	바이너리 읽기/쓰기 모드 (string이 string으로 변환되지 않음). FileOpen()에서 플래그 사용.	FileOpen
FILE_COMMON	모든 클라이언트 터미널의 공통 폴도에 있는 파일 경로 \Terminal\Common\Files. FileOpen(), FileCo	FileOpen, FileCopy, FileMove, FileIsExist

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	py(), FileMove() 및 FileIsExist() 기능에서 플래그 사용.	
FILE_CREATE_DATE	생성일	FileGetInteger
FILE_CSV	CSV 파일(모든 요소가 유니코드 또는 ANSI 형식의 문자열로 변환되고 분리자로 구별됨). FileOpen()에서 플래그 사용.	FileOpen
FILE_END	파일 끝 기호 가져오기	FileGetInteger
FILE_EXISTS	존재 여부 확인	FileGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
FILE_IS_ANSI	파일이 ANSI로 열립니다 (FILE_ANSI 참조)	FileGetInteger
FILE_IS_BINARY	파일이 바이너리 파일로 열립니다 (FILE_BIN 참조)	FileGetInteger
FILE_IS_COMMON	파일이 모든 터미널의 공유 폴더에 열립니다 (FILE_COMMON 참조)	FileGetInteger
FILE_IS_CSV	파일이 CSV로 열립니다 (FILE_CSV 참조)	FileGetInteger
FILE_IS_READABLE	열린 파일을 읽을 수 있습니다	FileGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	(FILE_READ 참조)	
FILE_IS_TEXT	텍스트 파일로 파일이 열립니다 (FILE_TXT 참조)	FileGetInteger
FILE_IS_WRITABLE	열린 파일을 쓸 수 있습니다 (FILE_WRITE 참조)	FileGetInteger
FILE_LINE_END	줄 끝 부호 가져오기	FileGetInteger
FILE_MODIFY_DATE	최종 수정일	FileGetInteger
FILE_POSITION	파일의 포인터 포지션	FileGetInteger
FILE_READ	파일이 읽기 위해 열렸습니다. FileOpen()에서 플래그	FileOpen

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	사용. FILE_WRITE 및 FILE_READ 파일 사양을 열기가 필요할 때.	
FILE_REWRITE	FileCopy() 및 FileMove() 기능을 사용하여 파일을 다시 쓸 수 있습니다. 파일이 있거나 쓰기 위해 열려 있어야 하며, 그렇지 않으면 파일이 열리지 않습니다.	FileCopy, FileMove
FILE_SHARE_READ	여러 프로그램에서	FileOpen

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	읽을 수 있는 공유 액세서. 플래그는 FileOpen()에서 사용되지만, 파일을 열 때 FILE_WRITE 및 FILE_READ 플래그를 표시할 필요는 없습니다.	
FILE_SHARE_WRITE	여러 프로그램에서 쓰기를 위한 공유 액세스. 플래그는 FileOpen()에서 사용되지만, 파일을 열 때 FILE_WRIT	FileOpen

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	E 및 FILE_READ 플래그를 표시할 필요는 없습니다.	
FILE_SIZE	파일 크기(바이트)	FileGetInteger
FILE_TXT	단순 텍스트 파일(csv 파일과 동일하지만 분리자를 고려하지 않음). FileOpen()에서 플래그 사용.	FileOpen
FILE_UNICODE	UNICODE 유형 문자열(2 바이트 심볼). FileOpen()에서 플래그 사용.	FileOpen

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
FILE_WRITE	파일이 쓰기 위해 열렸습니다. FileOpen()에서 플래그 사용. FILE_WRITE 및 FILE_READ 파일 사양을 열기가 필요할 때.	FileOpen
FLT_DIG	float 유형 유효 소수 자릿수	숫자 유형 상수
FLT_EPSILON	조건을 충족하는 최소값: 1.0+DBL_EPSILON != 1.0 (for float type)	숫자 유형 상수
FLT_MANT_DIG	float 유형 가수의 비트 수	숫자 유형 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
FLT_MAX	float 유형으로 나타낼 수 있는 최대 값	숫자 유형 상수
FLT_MAX_10_EXP	float 유형에 대한 지수 등급의 최대 십진수 값	숫자 유형 상수
FLT_MAX_EXP	float 유형에 대한 지수 등급의 최대 이진수 값	숫자 유형 상수
FLT_MIN	float 유형으로 나타낼 수 있는 최소 값(양수)	숫자 유형 상수
FLT_MIN_10_EXP	float 유형에 대한 지수 등급의 최소 십진수 값	숫자 유형 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
FLT_MIN_EXP	float 유형에 대한 지수 등급의 최소 이진수 값	숫자 유형 상수
FRIDAY	금요일	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
GANN_DOWN_TREND	하향 추세에 해당하는 선	ObjectSetInteger, ObjectGetInteger
GANN_UP_TREND	상승 추세에 해당하는 선	ObjectSetInteger, ObjectGetInteger
GATORJAW_LINE	턱 라인	지표선
GATORLIPS_LINE	립 선	지표선
GATORTEETH_LINE	톱니 선	지표선
IDABORT	"중단" 버튼이 눌렸습니다	MessageBox
IDCANCEL	"취소" 버튼이 눌렸습니다	MessageBox
IDCONTINUE	"계속" 버튼이 눌렸습니다	MessageBox

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
IDIGNORE	"무시하기" 버튼이 눌렸습니다	MessageBox
IDNO	"아니오" 버튼이 눌렸습니다	MessageBox
IDOK	"OK" 버튼이 눌렸습니다	MessageBox
IDRETRY	"재시도" 버튼이 눌렸습니다	MessageBox
IDTRYAGAIN	"다시 시도하기" 버튼이 눌렸습니다	MessageBox
IDYES	"예" 버튼이 눌렸습니다	MessageBox
IND_AC	Accelerator Oscillator	IndicatorCreate, IndicatorParameters
IND_AD	Accumulation/Distribution	IndicatorCreate, IndicatorParameters

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
IND_ADX	Average Directional Index	IndicatorCreate, IndicatorParameters
IND_ADXW	ADX by Welles Wilder	IndicatorCreate, IndicatorParameters
IND_ALLIGATOR	Alligator	IndicatorCreate, IndicatorParameters
IND_AMA	Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_AO	Awesome Oscillator	IndicatorCreate, IndicatorParameters
IND_ATR	Average True Range	IndicatorCreate, IndicatorParameters
IND_BANDS	Bollinger Bands®	IndicatorCreate, IndicatorParameters
IND_BEARS	Bears Power	IndicatorCreate, IndicatorParameters
IND_BULLS	Bulls Power	IndicatorCreate, IndicatorParameters
IND_BWMFI	Market Facilitation Index	IndicatorCreate, IndicatorParameters
IND_CCI	Commodity	IndicatorCreate, IndicatorParameters

© 2000–2025, MetaQuotes Ltd.

상수	설명	사용법
	Channel Index	
IND_CHAIKIN	Chaikin Oscillator	IndicatorCreate, IndicatorParameters
IND_CUSTOM	Custom indicator	IndicatorCreate, IndicatorParameters
IND_DEMA	Double Exponential Moving Average	IndicatorCreate, IndicatorParameters
IND_DEMARKER	DeMarker	IndicatorCreate, IndicatorParameters
IND_ENVELOPES	Envelopes	IndicatorCreate, IndicatorParameters
IND_FORCE	Force Index	IndicatorCreate, IndicatorParameters
IND_FRACTALS	Fractals	IndicatorCreate, IndicatorParameters
IND_FRAMA	Fractal Adaptive Moving Average	IndicatorCreate, IndicatorParameters
IND_GATOR	Gator Oscillator	IndicatorCreate, IndicatorParameters
IND_ICHIMOKU	Ichimoku Kinko Hyo	IndicatorCreate, IndicatorParameters

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
IND_MA	Moving Average	IndicatorCreate, IndicatorParameters
IND_MACD	MACD	IndicatorCreate, IndicatorParameters
IND_MFI	Money Flow Index	IndicatorCreate, IndicatorParameters
IND_MOMENTUM	Momentum	IndicatorCreate, IndicatorParameters
IND_OBV	On Balance Volume	IndicatorCreate, IndicatorParameters
IND_OSMA	OsMA	IndicatorCreate, IndicatorParameters
IND_RSI	Relative Strength Index	IndicatorCreate, IndicatorParameters
IND_RVI	Relative Vigor Index	IndicatorCreate, IndicatorParameters
IND_SAR	Parabolic SAR	IndicatorCreate, IndicatorParameters
IND_STDDEV	Standard Deviation	IndicatorCreate, IndicatorParameters
IND_STOCHASTIC	Stochastic Oscillator	IndicatorCreate, IndicatorParameters
IND_TEMA	Triple Exponential Moving	IndicatorCreate, IndicatorParameters

© 2000–2025, MetaQuotes Ltd.

상수	설명	사용법
	Average	
IND_TRIX	Triple Exponential Moving Averages Oscillator	IndicatorCreate, IndicatorParameters
IND_VIDYA	Variable Index Dynamic Average	IndicatorCreate, IndicatorParameters
IND_VOLUMES	Volumes	IndicatorCreate, IndicatorParameters
IND_WPR	Williams' Percent Range	IndicatorCreate, IndicatorParameters
INDICATOR_CALCULATIONS	중간 계산을 위한 보조 버퍼	SetIndexBuffer
INDICATOR_COLOR_INDEX	색상	SetIndexBuffer
INDICATOR_DATA	작성할 데이터	SetIndexBuffer
INDICATOR_DIGITS	지표 값 도면 정확도	IndicatorSetInteger
INDICATOR_HEIGHT	지표 창의 고정 높이	IndicatorSetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	(전처리기 명령 #property indicator_height)	
INDICATOR_LEVELCOLOR	레벨선 색상	IndicatorSetInteger
INDICATOR_LEVELS	지표창의 레벨 숫자	IndicatorSetInteger
INDICATOR_LEVELSTYLE	레벨선 스타일	IndicatorSetInteger
INDICATOR_LEVELTEXT	레벨 설명	IndicatorSetString
INDICATOR_LEVELVALUE	레벨 값	IndicatorSetDouble
INDICATOR_LEVELWIDTH	레벨선 굵기	IndicatorSetInteger
INDICATOR_MAXIMUM	지표창 최대	IndicatorSetDouble
INDICATOR_MINIMUM	지표창 최소	IndicatorSetDouble
INDICATOR_SHORTNAME	지표 단축 이름	IndicatorSetString
INT_MAX	int 유형으로 나타낼 수 있는 최대값	숫자 유형 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
INT_MIN	int 유형으로 나타낼 수 있는 최소값	숫자 유형 상수
INVALID_HANDLE	잘못된 핸들	기타 상수
IS_DEBUG_MODE	mq5 프로그램이 디버그 모드에서 작동한다는 플래그	기타 상수
IS_PROFILE_MODE	mq5 프로그램이 프로파일링 모드에서 작동한다는 플래그	기타 상수
KIJUNSEN_LINE	Kijun-sen line	지표선
LICENSE_DEMO	마켓에서 제공하는 유료 제품 평가판 전략 테스터	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	에서만 작동합니다	
LICENSE_FREE	무제한 무료 버전	MQLInfoInteger
LICENSE_FULL	구매한 라이센스 버전에서는 최소 5개 이상의 활성화가 허용됩니다. 활성화 횟수는 판매자에 의해 지정됩니다 판매자가 허용되는 활성화 수를 늘릴 수 있습니다	MQLInfoInteger
LICENSE_TIME	기간 제한 라이센스가 있는 버전	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
LONG_MAX	long 유형으로 나타낼 수 있는 최대값	숫자 유형 상수
LONG_MIN	long 유형으로 나타낼 수 있는 최소값	숫자 유형 상수
LOWER_BAND	하한	지표선
LOWER_HISTOGRAM	하단 히스토그램	지표선
LOWER_LINE	결론	지표선
M_1_PI	1/pi	수학 상수
M_2_PI	2/pi	수학 상수
M_2_SQRTPI	2/sqrt(pi)	수학 상수
M_E	e	수학 상수
M_LN10	ln(10)	수학 상수
M_LN2	ln(2)	수학 상수
M_LOG10E	log10(e)	수학 상수
M_LOG2E	log2(e)	수학 상수
M_PI	pi	수학 상수
M_PI_2	pi/2	수학 상수
M_PI_4	pi/4	수학 상수
M_SQRT1_2	1/sqrt(2)	수학 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
M_SQRT2	sqrt(2)	수학 상수
MAIN_LINE	본선	지표선
MB_ABORTRETRYIGNORE	메시지 창에는 중단, 재시도 및 무시하기의 세 가지 버튼이 있습니다	MessageBox
MB_CANCELTRYCONTINUE	메시지 창에는 취소, 다시 시도하기, 계속의 세 가지 버튼이 있습니다	MessageBox
MB_DEFBUTTON1	MB_DEFBUTTON2, MB_DEFBUTTON3, 또는 MB_DEFBUTTON4 버튼이 지정되지 않은 경	MessageBox

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	우, 첫 번째 버튼 MB_DEFBUTTON1 이 기본값 입니다	
MB_DEFBUTTON2	두 번째 버튼이 기본값입니다	MessageBox
MB_DEFBUTTON3	세 번째 버튼이 기본값입니다	MessageBox
MB_DEFBUTTON4	네 번째 버튼이 기본값입니다	MessageBox
MB_ICONEXCLAMATION, MB_ICONWARNING	느낌표/경고 기호 아이콘	MessageBox
MB_ICONINFORMATION, MB_ICONASTERISK	동그라미 친 기호	MessageBox
MB_ICONQUESTION	물음표 기호 아이콘	MessageBox
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	STOP 기호	MessageBox

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	아이콘	
MB_OK	메시지 창에는 OK 버튼이 하나만 있습니다. 기본값	MessageBox
MB_OKCANCEL	메시지 창에는 OK 및 취소의 두 가지 버튼이 있습니다	MessageBox
MB_RETRYCANCEL	메시지 창에는 재시도 및 취소 두 가지 버튼이 있습니다	MessageBox
MB_YESNO	메시지 창에는 예 및 아니오 두 가지 버튼이 있습니다	MessageBox

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
MB_YESNOCANCEL	메시지 창에는 예, 아니오 및 취소의 세 가지 버튼이 있습니다	MessageBox
MINUSDI_LINE	Line - DI	지표선
MODE_EMA	지수 평균	다듬는 방법
MODE_LWMA	선형 가중 평균	다듬는 방법
MODE_SMA	단순 평균	다듬는 방법
MODE_SMMA	평활 평균	다듬는 방법
MONDAY	월요일	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
MQL_DEBUG	디버그 모드를 나타내는 플래그	MQLInfoInteger
MQL_DLLS_ALLOWED	실행된 특정 프로그램에 대해 DLL을 사용할 수 있는	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	사용 권한	
MQL_FRAME_MODE	엑스퍼트 어드바이저가 다음으로 작동 중임을 나타내는 플래그: 최적화 결과 프레임 수집 모드	MQLInfoInteger
MQL_LICENSE_TYPE	EX5 모듈의 라이센스 유형. 라이센스는 MQLInfoInteger(MQL_LICENSE_TYPE)를 사용하여 요청을 수행하는 EX5 모듈을 참조합니다.	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
MQL_MEMORY_LIMIT	MQL5 프로그램에 대해 가능한 최대 동적 메모리 양 (MB)	MQLInfoInteger
MQL_MEMORY_USED	MQL5 프로그램에서 사용하는 메모리 크기 (MB)	MQLInfoInteger
MQL_OPTIMIZATION	최적화 프로세스를 나타내는 플래그	MQLInfoInteger
MQL_PROFILER	코드 프로파일링 모드에서 작동하는 프로그램을 나타내는 플래그	MQLInfoInteger
MQL_PROGRAM_NAME	실행된 mql5	MQLInfoString

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	프로그램 이름	
MQL_PROGRAM_PATH	실행된 특정 프로그램 경로	MQLInfoString
MQL_PROGRAM_TYPE	mql5 프로그램 유형	MQLInfoInteger
MQL_SIGNALS_ALLOWED	실행된 특정 프로그램에 대한 신호를 수정할 수 있는 권한	MQLInfoInteger
MQL_TESTER	테스터 프로세스를 나타내는 플래그	MQLInfoInteger
MQL_TRADE_ALLOWED	실행된 특정 프로그램과 거래할 수 있는 권한	MQLInfoInteger
MQL_VISUAL_MODE	비쥬얼 테스터	MQLInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	프로세스를 나타내는 플래그	
NULL	모든 유형에 대해 0	기타 상수
OBJ_ALL_PERIODS	모든 기간에 개체가 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_ARROW	화살표	개체 유형
OBJ_ARROW_BUY	구매 기호	개체 유형
OBJ_ARROW_CHECK	확인 기호	개체 유형
OBJ_ARROW_DOWN	아래 방향 화살표	개체 유형
OBJ_ARROW_LEFT_PRICE	좌측 가격 라벨	개체 유형
OBJ_ARROW_RIGHT_PRICE	우측 가격 라벨	개체 유형
OBJ_ARROW_SELL	판매 기호	개체 유형
OBJ_ARROW_STOP	중지 기호	개체 유형
OBJ_ARROW_THUMB_DOWN	엄지 아래로	개체 유형

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
OBJ_ARROW_THUMB_UP	엄지 위로	개체 유형
OBJ_ARROW_UP	위 방향 화살표	개체 유형
OBJ_ARROWED_LINE	화살표 선	개체 유형
OBJ_BITMAP	비트맵	개체 유형
OBJ_BITMAP_LABEL	비트맵 라벨	개체 유형
OBJ_BUTTON	버튼	개체 유형
OBJ_CHANNEL	등거리 채널	개체 유형
OBJ_CHART	차트	개체 유형
OBJ_CYCLES	싸이클 선	개체 유형
OBJ_EDIT	편집	개체 유형
OBJ_ELLIOTWAVE3	엘리엇 조정 파동	개체 유형
OBJ_ELLIOTWAVE5	엘리엇 동인 파동	개체 유형
OBJ_ELLIPSE	타원	개체 유형
OBJ_EVENT	이코노믹 캘린더의 이벤트에 해당하는 "이벤트" 개체	개체 유형

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
OBJ_EXPANSION	피보나치 확장	개체 유형
OBJ_FIBO	피보나치 되돌림	개체 유형
OBJ_FIBOARC	피보나치 호	개체 유형
OBJ_FIBOCHANNEL	피보나치 채널	개체 유형
OBJ_FIBOFAN	피보나치 팬	개체 유형
OBJ_FIBOTIMES	피보나치 시간대	개체 유형
OBJ_GANNFAN	Gann 팬	개체 유형
OBJ_GANNGRID	Gann 격자	개체 유형
OBJ_GANNLINE	Gann 선	개체 유형
OBJ_HLINE	수평선	개체 유형
OBJ_LABEL	라벨	개체 유형
OBJ_NO_PERIODS	개체가 모든 기간에 그려지는 것이 아닙니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_D1	개체가 일 차트	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	로 그려집니다	
OBJ_PERIOD_H1	개체가 1시간 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H12	개체가 12시간 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H2	개체가 2시간 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H3	개체가 3시간 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H4	개체가 4시간 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H6	개체가 6시간 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
OBJ_PERIOD_H8	개체가 8시간 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M1	개체가 1분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M10	개체가 10분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M12	개체가 12분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M15	개체가 15분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M2	개체가 2분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M20	개체가 20분 차트로 그려	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	집니다	
OBJ_PERIOD_M3	개체가 3분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M30	개체가 30분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M4	개체가 4분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M5	개체가 5분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M6	개체가 6분 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_MN1	개체가 월 차트로 그려집니다	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_W1	개체가 주 차트로 그	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	려집니다	
OBJ_PITCHFORK	앤드류의 피치포크 (Andrews' Pitchfork)	개체 유형
OBJ_RECTANGLE	직사각형	개체 유형
OBJ_RECTANGLE_LABEL	사용자 정의 그래픽 인터페이스를 만들고 설계하기 위한 "직사각형 레이블" 개체	개체 유형
OBJ_REGRESSION	선형 회귀 채널	개체 유형
OBJ_STDDEVCHANNEL	표준 편차 채널	개체 유형
OBJ_TEXT	텍스트	개체 유형
OBJ_TREND	추세선	개체 유형
OBJ_TRENDBYANGLE	각도별 추세선	개체 유형
OBJ_TRIANGLE	삼각형	개체 유형

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
OBJ_VLINE	수직선	개체 유형
OBJPROP_ALIGN	"편집" 개체의 수평 텍스트 정렬 (OBJ_EDIT)	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANCHOR	그래픽 개체의 고정점 위치	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANGLE	각도. 프로그램에서 생성된 각도가 지정되지 않은 개체의 경우 값은 EMPTY_VALUE와 같습니다	ObjectSetDouble, ObjectGetDouble
OBJPROP_ARROWCODE	화살표 개체의 화살표 코드	ObjectSetInteger, ObjectGetInteger
OBJPROP_BACK	배경의 개체	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
OBJPROP_BGCOLOR	OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL의 배경색	ObjectSetInteger, ObjectGetInteger
OBJPROP_BMPFILE	비트맵 라벨에 대한 BMP 파일의 이름. 추가 참조: 리소스	ObjectSetString, ObjectGetString
OBJPROP_BORDER_COLOR	OBJ_EDIT 및 OBJ_BUTTON 개체의 테두리 색	ObjectSetInteger, ObjectGetInteger
OBJPROP_BORDER_TYPE	"직사각형 라벨" 개체의 테두리 유형	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_ID	"차트" 개체의 ID (OBJ_CHART). It allows working	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	with the properties of this object like with a normal chart using the functions described in 차트 작업 에 설명된 기능을 사용하여 일반 차트와 같이 이 개체의 속성을 사용할 수 있지만, 일부 예외는 있습니다.	
OBJPROP_CHART_SCALE	차트 개체에 대한 스케일	ObjectSetInteger, ObjectGetInteger
OBJPROP_COLOR	색상	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
OBJPROP_CORNER	그래픽 개체를 연결할 차트의 모서리	ObjectSetInteger, ObjectGetInteger
OBJPROP_CREATETIME	개체 생성 시간	ObjectSetInteger, ObjectGetInteger
OBJPROP_DATE_SCALE	차트 개체의 시간 스케일 표시	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEGREE	엘리엇 파동 마킹 레벨	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEVIATION	표준 편차 채널에 대한 편차	ObjectSetDouble, ObjectGetDouble
OBJPROP_DIRECTION	Gann 개체의 추세	ObjectSetInteger, ObjectGetInteger
OBJPROP_DRAWLINES	엘리엇 파동 마킹 선 표시	ObjectSetInteger, ObjectGetInteger
OBJPROP_ELLIPSE	피보나치 호 개체의 전체 타원 표시	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	(OBJ_FIBOARC)	
OBJPROP_FILL	개체 색상 채우기 (OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)	ObjectSetInteger, ObjectGetInteger
OBJPROP_FONT	폰트	ObjectSetString, ObjectGetString
OBJPROP_FONTSIZE	폰트 크기	ObjectSetInteger, ObjectGetInteger
OBJPROP_HIDDEN	터미널 메뉴의 "차트" – "개체" – "개체 목록"에서 개체 목록에 그래픽 개체 이	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	름을 표시하는 것을 금지. true 값을 사용하면 목록에서 개체를 숨길 수 있습니다, 기본적으로 true는 캘린더 이벤트, 트레이딩 내역을 표시하는 개체 및 MQL5 프로그램에서 생성된 개체로 설정됩니다. 이러한 그래픽 개체를 보고 속성에 액세스하려	

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	면 "개체 목록" 창에서 "전체" 버튼을 클릭하십시오.	
OBJPROP_LEVELCOLOR	선 레벨 색상	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELS	레벨의 숫자	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELSTYLE	선 레벨 스타일	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELTEXT	레벨 설명	ObjectSetString, ObjectGetString
OBJPROP_LEVELVALUE	레벨 값	ObjectSetDouble, ObjectGetDouble
OBJPROP_LEVELWIDTH	선 레벨 굵기	ObjectSetInteger, ObjectGetInteger
OBJPROP_NAME	오브젝트 이름	ObjectSetString, ObjectGetString
OBJPROP_PERIOD	차트 개체에 대한 시간 프레임	ObjectSetInteger, ObjectGetInteger
OBJPROP_PRICE	가격 좌표	ObjectSetDouble, ObjectGetDouble
OBJPROP_PRICE_SCALE	차트 개체에 대한 가격 스	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	케일 표시	
OBJPROP_RAY	수직선은 차트의 모든 창을 통과합니다	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_LEFT	레이가 좌측으로 갑니다	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_RIGHT	레이가 오른쪽으로 갑니다	ObjectSetInteger, ObjectGetInteger
OBJPROP_READONLY	편집 개체에서 텍스트를 편집하는 기능	ObjectSetInteger, ObjectGetInteger
OBJPROP_SCALE	스케일 (Gann 개체 및 피보나치 호의 속성)	ObjectSetDouble, ObjectGetDouble
OBJPROP_SELECTABLE	개체 가용성	ObjectSetInteger, ObjectGetInteger
OBJPROP_SELECTED	개체가 선택되	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	었습니다	
OBJPROP_STATE	버튼 상태 (누름 / 누름)	ObjectSetInteger, ObjectGetInteger
OBJPROP_STYLE	스타일	ObjectSetInteger, ObjectGetInteger
OBJPROP_SYMBOL	차트 개체에 대한 심볼	ObjectSetString, ObjectGetString
OBJPROP_TEXT	개체 설명 (개체에 포함된 텍스트)	ObjectSetString, ObjectGetString
OBJPROP_TIME	시간 좌표	ObjectSetInteger, ObjectGetInteger
OBJPROP_TIMEFRAMES	시간 프레임에서 개체의 가시성	ObjectSetInteger, ObjectGetInteger
OBJPROP_TOOLTIP	도구 설명의 텍스트. 속성이 설정되어 있지 않으면 터미널에서 자동으	ObjectSetString, ObjectGetString

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	로 생성된 도구 설명이 표시됩니다. 도구 설명에 "\n" (라인 피드) 값을 설정하여 도구 설명을 비활성화할 수 있습니다	
OBJPROP_TYPE	개체 유형	ObjectSetInteger, ObjectGetInteger
OBJPROP_WIDTH	선 굵기	ObjectSetInteger, ObjectGetInteger
OBJPROP_XDISTANCE	바인딩 모서리에 X축을 따라가는 픽셀단위 거리(참고 참조)	ObjectSetInteger, ObjectGetInteger
OBJPROP_XOFFSET	그래픽 개체 "비트맵 라벨" 및 "비	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	트맵"에서 직사각형 가시 영역의 좌측 상단 모서리의 X 좌표 (OBJ_BITMAP_LABEL 및 OBJ_BITMAP). 값은 원본 이미지의 좌측 상단 모서리를 기준으로 픽셀단위로 설정됩니다.	
OBJPROP_XSIZE	X축을 따라 가는 개체의 너비(픽셀)입니다. OBJ_LABEL (읽기 전용), OBJ_BUTTO	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	N, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL 개체에 대해 지정됨.	
OBJPROP_YDISTANCE	바인딩 코너에서 Y축을 따라가는 거리 (참고참조)	ObjectSetInteger, ObjectGetInteger
OBJPROP_YOFFSET	그래픽 개체 "비트맵 라벨" 및 "비트맵"에서 직사각형 가시 영역의 좌측 상단 모서	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	리의 Y 좌표 (OBJ_BITMAP_LABEL 및 OBJ_BITMAP). 값은 원본 이미지의 좌측 상단 모서리를 기준으로 픽셀단위로 설정됩니다.	
OBJPROP_YSIZE	Y축을 따라 가는 개체의 높이(픽셀). OBJ_LABEL (읽기 전용), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL,	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	OBJ_EDIT, OBJ_RECTANGLE_LABEL 개체에 대해 지정됨.	
OBJPROP_ZORDER	차트 클릭 이벤트를 수신하기 위한 그래픽 개체의 우선순위 (CHARTEVENT_CLICK). 개체를 생성할 때 기본값인 0일 설정되며, 필요한 경우 우선 순위를 늘릴 수 있습니다. 개체를 서로 간에 적용할 때	ObjectSetInteger, ObjectGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	우선순위가 가장 높은 개체 중 하나만 CHARTEVENT_CLICK 이벤트를 수신합니다.	
ORDER_COMMENT	주문 코멘트	OrderGetString, HistoryOrderGetString
ORDER_FILLING_FOK	이 채우기 정책은 지정된 양으로만 주문을 채울 수 있음을 의미합니다. 현재 시장에서 필요한 금융상품을 구할 수 없는 경우 주문이 실행되지	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	않습니다. 현재 시중에 나와있는 몇 가지 제안을 이용해 필요한 수량을 채울 수 있습니다.	
ORDER_FILLING_IOC	이 모드는 트레이더가 주문에 표시된 범위 내에서 시장에서 최대 가용 볼륨으로 거래를 실행하기로 동의함을 의미합니다. 주문의 전체 볼륨을 채울 수	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	없는 경우 주문의 사용 가능한 볼륨이 채워지고 나머지 볼륨이 취소됩니다.	
ORDER_FILLING_RETURN	이 정책은 시장 주문(ORDER_TYPE_BUY 및 ORDER_TYPE_SELL), 제한 및 제한 중지(ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT 및 ORDER_TYP	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	E_SELL_STOP_LIMIT) 주문에만 사용되며, 마켓 또는 교환이 실행되는 심볼에만 사용됩니다. 시장 또는 제한 주문을 잔여 볼륨으로 일부 메우는 경우 취소되지 않고 추가 처리됩니다. ORDER_TYPE_BUY_STOP_LIMIT 및 ORDER_TYPE_SELL_STOP_LIMIT 주문	

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	을 활성화하기 위해 ORDER_FILLING_RETURN 유형의 해당 제한 주문 ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT 이 생성됩니다.	
ORDER_MAGIC	주문을 발주한 엑스퍼트 어드바이저의 ID (각 엑스퍼트 어드바이저가 고유한 번호를 매기도록 고안됨)	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ORDER_POSITION_ID	포지션 식별자는 실행 즉시 주문으로 설정됩니다 that is set to an order as soon as it is executed. 실행된 각 준문은 이미 존재하는 포지션을 열거나 수정하는 거래를 초래합니다. 정확히 이 포지션의 식별자는 현재 실행된 순서로 설정됩니다.	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ORDER_PRICE_CURRENT	주문 심볼의 현재 가격	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_OPEN	주문에 지정된 가격	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_STOPLIMIT	StopLimit 주문의 Limit 주문 가격	OrderGetDouble, HistoryOrderGetDouble
ORDER_SL	정지 손실 값	OrderGetDouble, HistoryOrderGetDouble
ORDER_STATE	주문 상태	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_CANCELED	클라이언트에 의해 주문이 취소됨	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_EXPIRED	주문 만료	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_FILLED	주문 완료	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PARTIAL	주문이 부분적으로 실행됨	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PLACED	주문 수락	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REJECTED	주문 거절	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ORDER_STATE_REQUEST_ADD	주문이 등록되는 중(트레이딩 시스템에 위치)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_CANCEL	주문이 삭제되는 중(트레이딩 시스템에서 삭제)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	주문이 수정되는 중(매개변수 변경)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_STARTED	주문이 확인되었지만, 브로커에 의해 아직 수락되지 않았습니다	OrderGetInteger, HistoryOrderGetInteger
ORDER_SYMBOL	주문의 심볼	OrderGetString, HistoryOrderGetString
ORDER_TIME_DAY	현재 거래일 주문까	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	지 양호	
ORDER_TIME_DONE	주문 실행 또는 취소 시간	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE_MSC	1970 년 1월 1일 이후 주문 실행/취소 시간(밀리초)	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	주문 만료 시간	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_GTC	주문 취소 까지 양호	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP	주문 설정 시간	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP_MSC	1970 년 1월 1일 이후 실행 주문 시간(밀리초)	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED	주문 만료 시까지 양호	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	주문은 지정된 날 23:59	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	:59까지 유효합니다. 이 시간이 트레이딩 세션을 벗어나면 주문이 가장 가까운 트레이딩 시간 내에 만료됩니다.	
ORDER_TP	수익 값 취하기	OrderGetDouble, HistoryOrderGetDouble
ORDER_TYPE	주문 유형	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY	마켓 구매 주문	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_LIMIT	구매 제한 보류 중인 주문	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	구매 중지 보류 중인 주문	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	주문 가격에 도달하면 보	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	류 중인 구매 제한 주문이 StopLimit 가격으로 배치됩니다	
ORDER_TYPE_FILLING	주문 채우기 유형	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL	마켓 판매 주문	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_LIMIT	판매 제한 보류 중인 주문	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP	판매 중지 보류 중인 주문	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP_LIMIT	주문 가격에 도달하면 보류 중인 판매 제한 주문이 StopLimit 가격으로 배치됩니다	OrderGetInteger, HistoryOrderGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
ORDER_TYPE_TIME	주문 수명	OrderGetInteger, HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	주문 현재 볼륨	OrderGetDouble, HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	주문 초기 볼륨	OrderGetDouble, HistoryOrderGetDouble
PERIOD_CURRENT	현재 시간 프레임	차트 시간 프레임
PERIOD_D1	1 일	차트 시간 프레임
PERIOD_H1	1 시간	차트 시간 프레임
PERIOD_H12	12 시간	차트 시간 프레임
PERIOD_H2	2 시간	차트 시간 프레임
PERIOD_H3	3 시간	차트 시간 프레임
PERIOD_H4	4 시간	차트 시간 프레임
PERIOD_H6	6 시간	차트 시간 프레임
PERIOD_H8	8 시간	차트 시간 프레임
PERIOD_M1	1 분	차트 시간 프레임
PERIOD_M10	10 분	차트 시간 프레임
PERIOD_M12	12 분	차트 시간 프레임
PERIOD_M15	15 분	차트 시간 프레임
PERIOD_M2	2 분	차트 시간 프레임
PERIOD_M20	20 분	차트 시간 프레임
PERIOD_M3	3 분	차트 시간 프레임
PERIOD_M30	30 분	차트 시간 프레임
PERIOD_M4	4 분	차트 시간 프레임
PERIOD_M5	5 분	차트 시간 프레임
PERIOD_M6	6 분	차트 시간 프레임
PERIOD_MN1	1 개월	차트 시간 프레임
PERIOD_W1	1 주	차트 시간 프레임

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
PLOT_ARROW	DRAW_ARROW 스타일의 화살표 코드	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_ARROW_SHIFT	DRAW_ARROW 스타일의 화살표 수직 이동	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_COLOR_INDEXES	색상의 수	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_BEGIN	그리기 없는 초기 막대 수 및 Data Window에서의 값	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_TYPE	그래픽 구성 유형	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_EMPTY_VALUE	드로잉이 없는 빈 플롯 값	PlotIndexSetDouble
PLOT_LABEL	Data Window에 표시할 지표 그래픽	PlotIndexSetString

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	시리즈 이름. 표시를 위해 여러 지표 버퍼가 필요한 복잡한 그래픽 스타일로 작업할 때는 ";"를 분리자로 사용하여 각 버퍼의 이름을 지정할 수 있습니다. 샘플코드는 다음에 나와있습니다: DRAW_CANDLES	
PLOT_LINE_COLOR	드로잉 색상을 포함하는 버퍼	PlotIndexSetInteger, PlotIndexGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	인덱스	
PLOT_LINE_STYLE	드로잉 선 스타일	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_WIDTH	드로잉 선의 굵기	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHIFT	막대형 시간 축을 따라 지표 플롯 이동	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHOW_DATA	Data Window에 구성값 표시 기호	PlotIndexSetInteger, PlotIndexGetInteger
PLUSDI_LINE	Line +DI	지표선
POINTER_AUTOMATIC	자동으로 생성된 개체 포인터 (new() 사용 안함)	CheckPointer
POINTER_DYNAMIC	new() 연산자에 의해 생성된 개체의	CheckPointer

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	포인터	
POINTER_INVALID	잘못된 포인터	CheckPointer
POSITION_COMMENT	포지션 코멘트	PositionGetString
POSITION_COMMISSION	수수료	PositionGetDouble
POSITION_IDENTIFIER	포지션 식별자는 새로 연 모든 포지션에 할당되며 포지션의 전체 수명 동안 변경되지 않습니다. 위치가 변하더라도 식별자는 변경되지 않습니다.	PositionGetInteger
POSITION_MAGIC	포지션 매직 넘버 (ORDER_MA	PositionGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	GIC참조)	
POSITION_PRICE_CURRENT	포지션 심볼의 현재 가격	PositionGetDouble
POSITION_PRICE_OPEN	포지션 오픈 가격	PositionGetDouble
POSITION_PROFIT	현재 이익	PositionGetDouble
POSITION_SL	열린 포지션의 손실 중지 레벨	PositionGetDouble
POSITION_SWAP	누적 스왑	PositionGetDouble
POSITION_SYMBOL	포지션의 심볼	PositionGetString
POSITION_TIME	포지션 오픈 시간	PositionGetInteger
POSITION_TIME_MSC	1970년 1월 1일 이후 포지션 오픈 시간(밀리초)	PositionGetInteger
POSITION_TIME_UPDATE	1970년 1월 1일 이후 포지션 변경	PositionGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	시간(초)	
POSITION_TIME_UPDATE_MSC	1970년 1월 1일 이후 위치 변경 시간(밀리초)	PositionGetInteger
POSITION_TP	열린 포지션의 이익 수준 달성	PositionGetDouble
POSITION_TYPE	포지션 유형	PositionGetInteger
POSITION_TYPE_BUY	구매	PositionGetInteger
POSITION_TYPE_SELL	판매	PositionGetInteger
POSITION_VOLUME	포지션 볼륨	PositionGetDouble
PRICE_CLOSE	마감 가격	가격 상수
PRICE_HIGH	기간의 최대 가격	가격 상수
PRICE_LOW	기간의 최소 가격	가격 상수
PRICE_MEDIAN	중간 값, (고가 + 저가)/2	가격 상수
PRICE_OPEN	오픈 가격	가격 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
PRICE_TYPICAL	표준 가격, (고가 + 저가 + 마감가)/3	가격 상수
PRICE_WEIGHTED	평균 가격, (고가 + 저가 + 마감가 + 마감가)/4	가격 상수
PROGRAM_EXPERT	엑스퍼트	MQLInfoInteger
PROGRAM_INDICATOR	지표	MQLInfoInteger
PROGRAM_SCRIPT	스크립트	MQLInfoInteger
REASON_ACCOUNT	계정 설정의 변경으로 인해 다른 계정이 활성화되었거나 거래 서버에 다시 연결되었습니다	UninitializeReason, OnDeinit
REASON_CHARTCHANGE	심볼 또는 차트 기간이 변경되	UninitializeReason, OnDeinit

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	었습니다	
REASON_CHARTCLOSE	차트가 닫혔습니다	UninitializeReason, OnDeinit
REASON_CLOSE	터미널이 닫혔습니다	UninitializeReason, OnDeinit
REASON_INITFAILED	이 값은 OnInit() 처리기가 0이 아닌 값을 반환했음을 의미합니다	UninitializeReason, OnDeinit
REASON_PARAMETERS	사용자가 입력 매개변수를 변경했습니다	UninitializeReason, OnDeinit
REASON_PROGRAM	엑스퍼트 어드바이저는 ExpertRemove() 기능을 호출하여 실행을	UninitializeReason, OnDeinit

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	종료하였습니다	
REASON_RECOMPILE	프로그램이 다시 컴파일되었습니다	UninitializeReason, OnDeinit
REASON_REMOVE	프로그램이 차트에서 삭제되었습니다	UninitializeReason, OnDeinit
REASON_TEMPLATE	새로운 템플릿이 적용되었습니다	UninitializeReason, OnDeinit
SATURDAY	토요일	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SEEK_CUR	파일 포인터의 현재 포지션	FileSeek
SEEK_END	파일 끝	FileSeek
SEEK_SET	파일 시작	FileSeek
SENKOUSPANA_LINE	센쿠(Senkou) 스	지표선

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	팬 A 선	
SENKOUSPANB_LINE	센쿠(Senkou) 스팬 B 선	지표선
SERIES_BARS_COUNT	현재 순간의 심볼 기간에 대한 막대 수	SeriesInfoInteger
SERIES_FIRSTDATE	현재 순간의 심볼 기간에 대한 첫 번째 날짜	SeriesInfoInteger
SERIES_LASTBAR_DATE	심볼 기간의 마지막 막대가 열린 시간	SeriesInfoInteger
SERIES_SERVER_FIRSTDATE	시간 프레임과 관계없이 서버의 심볼 내역에서 첫 번째 날짜	SeriesInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SERIES_SYNCHRONIZED	현재 순간의 심볼/가간 데이터 동기화 플래그	SeriesInfoInteger
SERIES_TERMINAL_FIRSTDATE	시간 프레임과 관계없이 클라이언트 터미널의 심볼 내역에서 첫 번째 날짜	SeriesInfoInteger
SHORT_MAX	short 유형으로 나타낼 수 있는 최대 값	숫자 유형 상수
SHORT_MIN	short 유형으로 나타낼 수 있는 최소 값	숫자 유형 상수
SIGNAL_BASE_AUTHOR_LOGIN	작성자 로그인	SignalBaseGetString
SIGNAL_BASE_BALANCE	계좌 잔고	SignalBaseGetDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SIGNAL_BASE_BROKER	브로커 이름 (회사)	SignalBaseGetString
SIGNAL_BASE_BROKER_SERVER	브로커 서버	SignalBaseGetString
SIGNAL_BASE_CURRENCY	시그널 기준 통화	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	발행일 (구독이 가능하게 되는 날짜)	SignalBaseGetInteger
SIGNAL_BASE_DATE_STARTED	모니터링 시작일	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	계정 지분	SignalBaseGetDouble
SIGNAL_BASE_GAIN	계정 획득	SignalBaseGetDouble
SIGNAL_BASE_ID	시그널 ID	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	계좌 레버리지	SignalBaseGetInteger
SIGNAL_BASE_MAX_DRAWDOWN	계정 최대 인출액	SignalBaseGetDouble
SIGNAL_BASE_NAME	시그널 이름	SignalBaseGetString
SIGNAL_BASE_PIPS	이익 (pips)	SignalBaseGetInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SIGNAL_BASE_PRICE	시그널 구독 가격	SignalBaseGetDouble
SIGNAL_BASE_RATING	포지션(율)	SignalBaseGetInteger
SIGNAL_BASE_ROI	투자 수익률(%)	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	구독자 수	SignalBaseGetInteger
SIGNAL_BASE_TRADE_MODE	계정 유형 (0–실제, 1–데모, 2–콘테스트)	SignalBaseGetInteger
SIGNAL_BASE_TRADES	거래 수	SignalBaseGetInteger
SIGNAL_INFO_CONFIRMATIONS_DISABLED	플래그가 확인 대화 상자 없이 동기화를 활성화합니다	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_COPY_SLTP	손실 중지 및 이익 획득 복사 플래그	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	예치율 (%)	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	지분 한도	SignalInfoGetDouble, SignalInfoSetDouble

© 2000–2025, MetaQuotes Ltd.

상수	설명	사용법
SIGNAL_INFO_ID	시그널 id, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	시그널 이름, r/o	SignalInfoGetString
SIGNAL_INFO_SLIPPAGE	하락 (포지션 동기화 및 거래 복사에서 시장 주문을 할 때 사용)	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_SUBSCRIPTION_ENABLED	"구독에 의한 거래 복사" 권한 플래그	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	"시그널 서비스 이용 약관에 동의" 플래그, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	사용된 최대 예치금 백분율 (%), r/o	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_LINE	시그널 라	지표선

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	인	
STAT_BALANCE_DD	통화 기준 최대 잔액 감소 트레이딩 과정에서 잔액은 여러 차례 하락할 수 있으며, 여기에서 가장 큰 가치를 취합니다	TesterStatistics
STAT_BALANCE_DD_RELATIVE	최대 잔액 감소 시점에 백분율로 기록된 통화 기준 잔액 감소 (STAT_BALANCE_DDREL_PERCENT).	TesterStatistics
STAT_BALANCE_DDREL_PERCENT	최대 잔액 감소율(%)	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	트레이딩 과정에서 잔액은 여러 차례 하락할 수 있으며, 각각 상대 하락 값(%)을 계산합니다 가장 큰 값이 반환됩니다	
STAT_BALANCEDD_PERCENT	통화 기준으로 최대 잔액 감소 시점에 기록된 비율로서의 잔액 감소 (STAT_BALANCE_DD).	TesterStatistics
STAT_BALANCEMIN	최소 잔액 값	TesterStatistics
STAT_CONLOSSMAX	연속 손실 거래	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	에서 최대 손실 값이 0 보다 작거나 같습니다	
STAT_CONLOSSMAX_TRADES	STAT_CONLOSSMAX 을 형성한 거래 수 (연속 손실 거래에서 최대 손실)	TesterStatistics
STAT_CONPROFITMAX	연속 수익 거래에서 최대 이익 값이 0 보다 크거나 같습니다	TesterStatistics
STAT_CONPROFITMAX_TRADES	STAT_CONPROFITMAX 를 형성한 거래 수 (연속 수익 거래에서 최	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	대 이익)	
STAT_CUSTOM_ONTESTER	OnTester() 함수로 반환된 계산된 사용자 지정 최적화 기준	TesterStatistics
STAT_DEALS	거래 수	TesterStatistics
STAT_EQUITY_DD	통화 기준 최대 최대 지분 감소. 트레이딩 과정에서 지분에 많은 하락이 나타날 수 있으며, 여기에서 가장 큰 값을 갖습니다	TesterStatistics
STAT_EQUITY_DD_RELATIVE	최대 지분 율 하락 시점에 기록	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	된 통화 기준 지분 감소율 (%) (STAT_EQUITY_DDREL_PERCENT).	
STAT_EQUITY_DDREL_PERCENT	백분율로서 최대 지분 감소 트레이딩 과정에서 지분은 여러 차례 하락할 수 있으며, 각각 상대 하락 값(%)을 계산합니다. 가장 큰 값이 반환됩니다	TesterStatistics
STAT_EQUITYDD_PERCENT	통화 기준으로 최대 지분 감소	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	시점에 기록된 비율 감소(STAT_EQUITY_DD).	
STAT_EQUITYMIN	최소 지분 값	TesterStatistics
STAT_EXPECTED_PAYOFF	예상 수익	TesterStatistics
STAT_GROSS_LOSS	총 손실, 모든 마이너스 거래의 합계 값이 0보다 작거나 같습니다	TesterStatistics
STAT_GROSS_PROFIT	총 수익, 모든 수익 거래(양수) 거래의 합산 값이 0보다 크거나 같습니다	TesterStatistics
STAT_INITIAL_DEPOSIT	초기 예치금 값	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
STAT_LONG_TRADES	장기 거래	TesterStatistics
STAT_LOSS_TRADES	손실 거래	TesterStatistics
STAT_LOSSTRADES_AVGCON	연속된 손실 거래의 평균 기간	TesterStatistics
STAT_MAX_CONLOSS_TRADES	가장 오랫동안 연속된 손실 거래의 거래 수 STAT_MAX_CONLOSSES	TesterStatistics
STAT_MAX_CONLOSSES	가장 오래 동안 연속된 손실 거래의 총 손실	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	가장 오랫동안 연속된 수익 거래의 거래 수 STAT_MAX_CONWINS	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
STAT_MAX_CONWINS	가장 오랫동안 연속된 수익 거래의 총 수익	TesterStatistics
STAT_MAX_LOSSTRADE	최대 손실 - 모든 손실 거래 중 가장 낮은 값. 값이 0보다 작거나 같습니다	TesterStatistics
STAT_MAX_PROFITTRADE	최대 이익 - 모든 수익 거래 중 가장 큰 값. 값이 0보다 크거나 같습니다	TesterStatistics
STAT_MIN_MARGINLEVEL	마진 레벨의 최소값	TesterStatistics
STAT_PROFIT	테스트 후 순이익, STAT_GRO	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	SS_PROFIT과 STAT_GROSS_LOSS의 합계 (STAT_GROSS_LOSS는 항상 0보다 작거나 같습니다)	
STAT_PROFIT_FACTOR	STAT_GROSS_PROFIT/STAT_GROSS_LOSS의 비율과 같은 수익률. STAT_GROSS_LOSS=0 이면, 이익 계수가 다음과 같습니다: DBL_MAX	TesterStatistics
STAT_PROFIT_LONGTRADES	수익성이 있는	TesterStatistics

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	장기 거래	
STAT_PROFIT_SHORTTRADES	수익성 있는 단기 거래	TesterStatistics
STAT_PROFIT_TRADES	수익 거래	TesterStatistics
STAT_PROFITTRADES_AVGCON	연속된 수익 거래의 평균 기간	TesterStatistics
STAT_RECOVERY_FACTOR	STAT_PROFIT/STAT_BALANCE_DD의 비율과 같은 복구 계수	TesterStatistics
STAT_SHARPE_RATIO	샤프 비율	TesterStatistics
STAT_SHORT_TRADES	단기 거래	TesterStatistics
STAT_TRADES	거래 수	TesterStatistics
STAT_WITHDRAWAL	계좌에서 인출된 돈	TesterStatistics
STO_CLOSECLOSE	마감/마감 가격을 기준으	가격 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	로 계산	
STO_LOWHIGH	저가/고가 가격을 기준으로 계산	가격 상수
STYLE_DASH	파선	그리기 스타일
STYLE_DASHDOT	일점쇄선	그리기 스타일
STYLE_DASHDOTDOT	이점쇄선	그리기 스타일
STYLE_DOT	점선	그리기 스타일
STYLE_SOLID	실선	그리기 스타일
SUNDAY	일요일	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SYMBOL_ASK	요청 – 가장 좋은 구매 제안	SymbolInfoDouble
SYMBOL_ASKHIGH	오늘의 최대 요청	SymbolInfoDouble
SYMBOL_ASKLOW	오늘의 최소 요청	SymbolInfoDouble
SYMBOL_BANK	현재 시세 피더	SymbolInfoString
SYMBOL_BASIS	파생상품의 기본 자산	SymbolInfoString

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SYMBOL_BID	입찰 – 가장 좋은 판매 제안	SymbolInfoDouble
SYMBOL_BIDHIGH	오늘의 최대 입찰	SymbolInfoDouble
SYMBOL_BIDLOW	오늘의 최소 입찰	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	CFD 모드 – CFD 이익 및 마진 계산	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	CFD 인덱스 모드 – 인덱스별 CFD 이익 및 마진 계산	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD 레버리지 모드 – 레버리지 트레이딩에서 CFD의 이익 및 마진 계산	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SYMBOL_CALC_MODE_EXCH_FUTURES	선물 모드 – 증권 거래의 선물 거래 계약의 이익 및 마진 계산	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS 선물 모드 – FORTS 선물 거래 계약 이익 및 마진 계산.	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_STOCKS	교환 모드 – 증권 거래의 이익 및 마진 계산	SymbolInfoInteger
SYMBOL_CALC_MODE_FOREX	외환 거래 모드 – 외환 거래의 이익 및 마진 계산	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	선물 모드 – 선물 이익 및 마	SymbolInfoInteger

© 2000–2025, MetaQuotes Ltd.

상수	설명	사용법
	진 계산	
SYMBOL_CALC_MODE_SERV_COLLATERAL	담보 모드 – 트레이딩 계정에서 거래할 수 없는 자산으로 심볼이 사용됩니다.	SymbolInfoInteger
SYMBOL_CURRENCY_BASE	심볼의 기본 통화	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	마진 통화	SymbolInfoString
SYMBOL_CURRENCY_PROFIT	수익 통화	SymbolInfoString
SYMBOL_DESCRIPTION	심볼 설명	SymbolInfoString
SYMBOL_DIGITS	소수점 아래 숫자	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	주문은 오늘까지 유효합니다	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	주문이 명시적으로 취소될 때까지 무제한 기간 동	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	안 유효합니다	
SYMBOL_EXPIRATION_MODE	허용된 주문만료 모드의 플래그	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED	만료 시간이 이 주문에 지정되어 있습니다	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED_DAY	만료 날짜가 주문에 지정되어 있습니다	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	심볼 거래 종료일 (일반적으로 선물에 사용됨)	SymbolInfoInteger
SYMBOL_FILLING_FOK	이 정책은 지정된 볼륨에 대해서만 거래를 실행할 수 있	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	음을 의미 합니 다. 현 재 시 장에 서 필 요한 금융 상품 을 구 할 수 없는 경우 주문 이 실 행되 지 않 습니 다. 현 재 시 중에 나와 있는 몇 가 지 제 안을 이용 해 필 요한 수량 을 채 울 수 있습 니다.	
SYMBOL_FILLING_IOC	이 경 우, 트 레이 더는 주문 서에 명시 된 범 위 내 에서 시장 에서 최대	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	한 이용 가능한 양으로 거래를 실행하는 데 동의합니다. 주문이 완전히 채워지지 않을 경우, 사용 가능한 주문량이 채워지고 나머지 주문량은 취소됩니다. IOC 주문 사용 가능성은 거래 서버에서 결정됩니다.	
SYMBOL_FILLING_MODE	주문 작성 모드 허용 플래그	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SYMBOL_ISIN	ISIN 시스템의 심볼 이름 (International Securities Identification Number). 국제 보안 식별 번호는 보안을 고유하게 식별하는 12 자리 영문숫자 코드입니다. 이 심볼 속성의 존재 여부는 거래 서버의 측면에서 결정됩니다	SymbolInfoString
SYMBOL_LAST	마지막 거래 가격	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SYMBOL_LASTHIGH	오늘 최대 마지막 거래	SymbolInfoDouble
SYMBOL_LASTLOW	오늘 최소 마지막 거래	SymbolInfoDouble
SYMBOL_MARGIN_INITIAL	최초 마진은 1로트로 된 포지션을 오프닝하는 데 필요한 마진 통화의 금액을 의미합니다. 클라이언트가 시장에 진입할 때 자신의 자산을 확인하는 데 사용됩니다.	SymbolInfoDouble
SYMBOL_MARGIN_MAINTENANCE	유지 마진. 설정된 경우 한	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	로트에서 충전된 심볼의 마진 통화로 마진 금액을 설정합니다. 클라이언트의 계정 상태가 변경될 때 자신의 자산을 확인하는 데 사용됩니다. 유지 마진 여유 도가 0이면 초기 마진이 사용됩니다.	
SYMBOL_OPTION_MODE	옵션 유형	SymbolInfoInteger
SYMBOL_OPTION_MODE_EUROPEAN	유러피언 옵션은 지정된 날짜(만료,	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	실행일, 배송일)에만 행사할 수 있습니다	
SYMBOL_OPTION_MODE_AMERICAN	아메리칸 옵션은 만료일 또는 그 전에 어느 거래일에나 행사할 수 있습니다. 구매자가 옵션을 행사할 수 있는 기간이 지정되었습니다	SymbolInfoInteger
SYMBOL_OPTION_RIGHT	옵션 우측(콜/풋)	SymbolInfoInteger
SYMBOL_OPTION_RIGHT_CALL	콜(Call) 옵션은 지정된 가격으로 자산	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	을 구입할 수 있는 권한을 제공합니다	
SYMBOL_OPTION_RIGHT_PUT	풋(Put) 옵션은 자산을 지정된 가격으로 판매할 수 있는 권한을 제공합니다	SymbolInfoInteger
SYMBOL_OPTION_STRIKE	옵션의 스트라이크 가격. 옵션 구매자가 기본 자산을 매입(콜 옵션)하거나 매도(풋 옵션)할 수 있는 가격이며 옵션 판매	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	자는 기본 자산의 적절한 금액을 매각하거나 매입해야 합니다.	
SYMBOL_ORDER_LIMIT	주문 제한 허용됨((구매 한도 및 판매 한도)	SymbolInfoInteger
SYMBOL_ORDER_MARKET	마켓 주문 허용됨(구매 및 판매)	SymbolInfoInteger
SYMBOL_ORDER_MODE	허용된 주문 유형의 플래그	SymbolInfoInteger
SYMBOL_ORDER_SL	손실 중지가 허용됨	SymbolInfoInteger
SYMBOL_ORDER_STOP	중지 주문 허용됨(구매 중지 및 판매 중지)	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SYMBOL_ORDER_STOP_LIMIT	제한 중지 주문 허용됨(구매 중지 제한 및 판매 중지 제한)	SymbolInfoInteger
SYMBOL_ORDER_TP	이익 취득이 허용됨	SymbolInfoInteger
SYMBOL_PATH	심볼 트리의 경로	SymbolInfoString
SYMBOL_POINT	심볼 포인트 값	SymbolInfoDouble
SYMBOL_SELECT	Market Watch에서 기호가 선택됨	SymbolInfoInteger
SYMBOL_SESSION_AW	현재 세션의 평균 가중 가격	SymbolInfoDouble
SYMBOL_SESSION_BUY_ORDERS	현재 구매 주문 수	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	현재 구매 주문 볼륨	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SYMBOL_SESSION_CLOSE	현재 세션 마감 가격	SymbolInfoDouble
SYMBOL_SESSION_DEALS	현재 세션 거래 수	SymbolInfoInteger
SYMBOL_SESSION_INTEREST	오픈 이자 요약	SymbolInfoDouble
SYMBOL_SESSION_OPEN	현재 세션 오픈 가격	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MAX	현재 세션 최대 가격	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MIN	현재 세션 최소 가격	SymbolInfoDouble
SYMBOL_SESSION_PRICE_SETTLEMENT	현재 세션 정산 가격	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	현재 판매 주문 수	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	현재 판매 주문 볼륨	SymbolInfoDouble
SYMBOL_SESSION_TURNOVER	현재 세션의 회수 요약	SymbolInfoDouble
SYMBOL_SESSION_VOLUME	현재 세션	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	거래 볼륨 요약	
SYMBOL_SPREAD	포인트 내 값 스프레드	SymbolInfoInteger
SYMBOL_SPREAD_FLOAT	플로팅 (floating) 스프레드 지표	SymbolInfoInteger
SYMBOL_START_TIME	심볼 거래가 시작된 날짜 (일반적으로 선물에 사용됨)	SymbolInfoInteger
SYMBOL_SWAP_LONG	장기 스왑 값	SymbolInfoDouble
SYMBOL_SWAP_MODE	스왑 계산 모델	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	스왑은 통화, 고객 예금 통화로 청구됩니다	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	스왑은 심볼의	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	마진 통화로 청구됩니다	
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	스왑은 기호의 기준 통화로 청구됩니다	SymbolInfoInteger
SYMBOL_SWAP_MODE_DISABLED	스왑 사용 안 함 (스왑 없음)	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_CURRENT	스왑은 스왑 계산 시 상품 가격에서 지정된 연이자로 청구됩니다 (포준 은행 연도는 360일)	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_OPEN	스왑은 포지션 오픈 가격에서 지정된 연이자	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	로 청구됩니다 (표준 은행 연도는 360 일)	
SYMBOL_SWAP_MODE_POINTS	스왑은 포인트 단위로 청구됩니다	SymbolInfoInteger
SYMBOL_SWAP_MODE_REOPEN_BID	스왑은 포지션을 다시 오픈하는 방식으로 청구됩니다. 트레이딩 일이 종료되면 해당 지점은 닫힙니다. 다음날 현재 입찰 가격 +/- 지정된 포인트 수 (매개변수	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	SYMBOL_SWAP_LONG 및 SYMBOL_SWAP_SHORT)로 다시 열립니다	
SYMBOL_SWAP_MODE_REOPEN_CURRENT	스왑은 포지션을 다시 오픈하는 방식으로 청구됩니다. 트레이딩 일이 종료되면 해당 지점은 닫힙니다. 다음날 마감 가격 +/- 지정된 포인트 수(매개변수 SYMBOL_SWAP_LONG 및	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	SYMBOL_SWAP_SHORT)로 다시 열립니다	
SYMBOL_SWAP_ROLLOVER3DAYS	스왑 롤오버를 3일 충전할 요일	SymbolInfoInteger
SYMBOL_SWAP_SHORT	단기 스왑 값	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	다음에 표시되는 최대 요청 수: Depth of Market. 요청 대기열이 없는 심볼의 값은 0입니다.	SymbolInfoInteger
SYMBOL_TIME	마지막 시세의 시간	SymbolInfoInteger
SYMBOL_TRADE_CALC_MODE	계약 가격 계산 모드	SymbolInfoInteger
SYMBOL_TRADE_CONTRACT_SIZE	거래 계약	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	규모	
SYMBOL_TRADE_EXECUTION_EXCHANGE	교환 실행	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	즉시 실행	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	마켓 실행	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_REQUEST	요청 실행	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	거래 실행 모드	SymbolInfoInteger
SYMBOL_TRADE_FREEZE_LEVEL	동결 거래 작업 거리 (포인트)	SymbolInfoInteger
SYMBOL_TRADE_MODE	주문 실행 유형	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	포지션 닫기 작업만 허용됨	SymbolInfoInteger
SYMBOL_TRADE_MODE_DISABLED	심볼에 대해 거래가 비활성화 되었습니다	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	거래 제한 없음	SymbolInfoInteger
SYMBOL_TRADE_MODE_LONGONLY	장기 포지션만	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	허용됨	
SYMBOL_TRADE_MODE_SHORTONLY	단기 포지션만 허용됨	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	현재 마감 가격에서 중지 주문까지 포인트 표시 최소화	SymbolInfoInteger
SYMBOL_TRADE_TICK_SIZE	최소 가격 변동	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	SYMBOL_TRADE_TICK_VALUE_PROFIT의 값	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_LOSS	손실 포지션에 대해 계산된 틱 가격	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_PROFIT	수익 포지션에 대해 계산된 틱 가격	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
SYMBOL_VOLUME	전 거래 볼륨	SymbolInfoInteger
SYMBOL_VOLUME_LIMIT	심볼에 대한 한 방향(구매 도는 판매)으로 보류 중인 주문과 오픈 포지션의 최대 총 허용 볼륨입니다. 예를 들어 5 로트의 경우, 5 로트의 오픈 구매 포지션과 5 로트의 보류 중인 주문 판매 한도를 지정할 수 있습니다. 그러나 이 경우	SymbolInfoDouble

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	한 방향의 총 볼륨이 제한을 초과하기 때문에 구매 한도를 보류 중인 주문이나 5개 이상의 로트 수로 판매 한도를 지정할 수 없습니다.	
SYMBOL_VOLUME_MAX	거래 최대 볼륨	SymbolInfoDouble
SYMBOL_VOLUME_MIN	거래 최소 볼륨	SymbolInfoDouble
SYMBOL_VOLUME_STEP	거래 실행 최소 볼륨 변경 단계	SymbolInfoDouble
SYMBOL_VOLUMEHIGH	최대 일 볼륨	SymbolInfoInteger
SYMBOL_VOLUMELOW	최소 일 볼륨	SymbolInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
TENKANSEN_LINE	텐칸센 (Tenkan-sen) 선	지표선
TERMINAL_BUILD	클라이언트 터미널 빌드 번호	TerminalInfoInteger
TERMINAL_CODEPAGE	클라이언트 터미널에 설치된 언어의 코드 페이지 수	TerminalInfoInteger
TERMINAL_COMMONDATA_PATH	컴퓨터에 설치된 모든 터미널의 공통 경로	TerminalInfoString
TERMINAL_COMMUNITY_ACCOUNT	이 플래그는 MQL5.커뮤니티 권한 부여 데이터가 있음을 나타냅니다	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
TERMINAL_COMMUNITY_BALANCE	MQL5.커뮤니티 내 잔고	TerminalInfoDouble
TERMINAL_COMMUNITY_CONNECTION	MQL5.커뮤니티로 연결	TerminalInfoInteger
TERMINAL_COMPANY	회사 이름	TerminalInfoString
TERMINAL_CONNECTED	거래 서버로 연결	TerminalInfoInteger
TERMINAL_CPU_CORES	시스템의 CPU 코어 수	TerminalInfoInteger
TERMINAL_DATA_PATH	터미널 데이터가 저장되는 폴더	TerminalInfoString
TERMINAL_DISK_SPACE	터미널(에이전트)의 MQL5\Files 폴더에 사용 가능한 디스크 공간 (MB)	TerminalInfoInteger
TERMINAL_DLLS_ALLOWED	DLL 사용	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	권한	
TERMINAL_EMAIL_ENABLED	터미널 설정에 지정된 SMTP 서버 및 로그인을 사용하여 이메일을 전송할 수 있는 사용 권한	TerminalInfoInteger
TERMINAL_FTP_ENABLED	터미널 설정에 지정된 FTP 서버 및 로그인을 사용하여 보고서를 전송할 수 있는 사용 권한	TerminalInfoInteger
TERMINAL_LANGUAGE	터미널의 언어	TerminalInfoString
TERMINAL_MAXBARS	최대 막대가 차트에	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	서 가운트 됩니다	
TERMINAL_MEMORY_AVAILABLE	터미널(에이전트) 프로세스의 사용 가능한 메모리 (MB)	TerminalInfoInteger
TERMINAL_MEMORY_PHYSICAL	시스템의 물리적 메모리 (MB)	TerminalInfoInteger
TERMINAL_MEMORY_TOTAL	터미널(에이전트) 프로세스에서 사용 가능한 메모리 (MB)	TerminalInfoInteger
TERMINAL_MEMORY_USED	터미널(에이전트)에서 사용하는 메모리 (MB)	TerminalInfoInteger
TERMINAL_MQID	이 플래그는 푸쉬 알	TerminalInfoInteger

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	림에 대한 MetaQuotes ID 데이터가 있음을 나타냅니다	
TERMINAL_NAME	터미널 이름	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	스마트폰에 알림 전송을 허가	TerminalInfoInteger
TERMINAL_OPENCL_SUPPORT	0x00010002 = 1.2 형식의 지원되는 OpenCL 버전 입니다. "0"은 OpenCL이 지원되지 않음을 나타냅니다	TerminalInfoInteger
TERMINAL_PATH	터미널이 시작되는 폴더	TerminalInfoString

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
TERMINAL_PING_LAST	거래 서버에 대한 핑(ping)의 알려진 마지막 값(마이크로초). 1초는 백만 마이크로 초로 구성됩니다	TerminalInfoInteger
TERMINAL_SCREEN_DPI	화면에 표시되는 정보 해상도는 DPI(인치 당 도트 라인)의 수로 측정됩니다.	TerminalInfoInteger
TERMINAL_TRADE_ALLOWED	거래 허가	TerminalInfoInteger
TERMINAL_X64	"64비트 터미널" 지표	TerminalInfoInteger
THURSDAY	목요일	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TRADE_ACTION_DEAL	지정된 매개변수(마	MqlTradeRequest

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	켓 주문)를 사용하여 즉시 실행할 수 있도록 거래 주문 작성	
TRADE_ACTION_MODIFY	이전에 발주한 주문의 매개변수 수정	MqlTradeRequest
TRADE_ACTION_PENDING	지정된 조건(보류 주문)에서 실행을 위한 거래 주문 작성	MqlTradeRequest
TRADE_ACTION_REMOVE	이전에 배치된 보류 중인 주문 삭제	MqlTradeRequest
TRADE_ACTION_SLTP	손실 중지 및 오픈 포지션의 이익 획	MqlTradeRequest

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	득 값 수정	
TRADE_RETCODE_CANCEL	트레이더에 의해 요청 취소됨	MqlTradeResult
TRADE_RETCODE_CLIENT_DISABLES_AT	클라이언트 터미널에 의해 자동 거래가 비활성화됨	MqlTradeResult
TRADE_RETCODE_CONNECTION	거래 서버에 연결되지 않음	MqlTradeResult
TRADE_RETCODE_DONE	요청 완료됨	MqlTradeResult
TRADE_RETCODE_DONE_PARTIAL	요청의 일부만 완료되었습니다	MqlTradeResult
TRADE_RETCODE_ERROR	요청 처리 오류	MqlTradeResult
TRADE_RETCODE_FROZEN	주문 또는 포지션이 동결됨	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
TRADE_RETCODE_INVALID	유효하지 않은 요청	MqlTradeResult
TRADE_RETCODE_INVALID_EXPIRATION	유효하지 않은 요청의 주문 만료 날짜	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	유효하지 않은 주문 작성 유형	MqlTradeResult
TRADE_RETCODE_INVALID_ORDER	잘못되거나 금지된 주문 유형	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	요청의 유효하지 않은 가격	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	요청의 유효하지 않은 중지	MqlTradeResult
TRADE_RETCODE_INVALID_VOLUME	요청의 유효하지 않은 볼륨	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	보류 중인	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	주문 수가 한계에 도달했습니다	
TRADE_RETCODE_LIMIT_VOLUME	심볼에 대한 주문 및 포지션의 볼륨이 한계에 도달했습니다	MqlTradeResult
TRADE_RETCODE_LOCKED	처리를 위해 요청 잠김	MqlTradeResult
TRADE_RETCODE_MARKET_CLOSED	마켓이 닫힘	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	요청 변경 없음	MqlTradeResult
TRADE_RETCODE_NO_MONEY	요청을 완료할 충분한 자금이 없습니다	MqlTradeResult
TRADE_RETCODE_ONLY_REAL	활성 계정에 대해서만 작업이 허용	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	됩니다	
TRADE_RETCODE_ORDER_CHANGED	주문 상태 변경됨	MqlTradeResult
TRADE_RETCODE_PLACED	주문 발주됨	MqlTradeResult
TRADE_RETCODE_POSITION_CLOSED	지정한 POSITION_IDENTIFIER 가 있는 위치가 이미 닫혔습니다	MqlTradeResult
TRADE_RETCODE_PRICE_CHANGED	청구된 가격	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	요청을 처리할 견적이 없습니다	MqlTradeResult
TRADE_RETCODE_REJECT	요청 거절됨	MqlTradeResult
TRADE_RETCODE_REQUOTE	재견적	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	서버에서 자동 트레이딩 사용 안함	MqlTradeResult

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
TRADE_RETCODE_TIMEOUT	시간 초과에 의해 요청이 취소됨	MqlTradeResult
TRADE_RETCODE_TOO_MANY_REQUESTS	너무 빈번한 요청	MqlTradeResult
TRADE_RETCODE_TRADE_DISABLED	거래가 비활성화되었습니다	MqlTradeResult
TRADE_TRANSACTION_DEAL_ADD	내역에 거래를 추가하는 것. 주문 실행 또는 계전 잔액으로 작업 수행의 결과로 작업이 수행됩니다.	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_DELETE	내역에서 거래를 삭제하는 중입니다. 이전에	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	실행된 거래가 서버에서 삭제되는 경우가 있을 수 있습니다. 예를 들어 거래가 이전에 브로커에 의해 이전에 외부 거래 시스템(거래소)에서 삭제되었습니다.	
TRADE_TRANSACTION_DEAL_UPDATE	내역에 거래를 업데이트 중. 이전에 실행된 거래가 서버에서 변경된 경우가 있을 수 있	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	습니다. 예를 들어 거래가 이전에 브로커에 의해 이전에 외부 거래 시스템(거래소)에서 변경되었습니다.	
TRADE_TRANSACTION_HISTORY_ADD	실행 또는 취소의 결과로 내역에 주문 추가.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_DELETE	주문 내역에서 주문을 삭제. 이 유형은 거래 서버 측의 기능 향상을 위해 제공됩니다.	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
TRADE_TRANSACTION_HISTORY_UPDATE	주문 내역에 있는 주문을 변경. 이 유형은 거래 서버 측의 기능 향상을 위해 제공됩니다.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_ADD	새로 열린 주문을 추가.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_DELETE	열려 있는 주문 목록에서 주문을 제거. 주문은 적절한 요청 또는 실행(채우기)을 설정하고 내역으로 이동하면 열려있는 주문에	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	서 삭제할 수 있습니다.	
TRADE_TRANSACTION_ORDER_UPDATE	열린 주문을 업데이트. 업데이트에는 클라이언트 터미널 또는 거래 서버 측의 명백한 변경뿐만 아니라 설정 시 주문 상태의 변경도 포함됩니다. (예: ORDER_STATE_STARTED에서 ORDER_STATE_PLACED로 또는 ORDE	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	R_STATE_PLACED에서 ORDER_STATE_PARTIAL로 전환 등).	
TRADE_TRANSACTION_POSITION	거래 실행과 관련이 없는 직책 변경. 이 트랜잭션 유형은 거래 서버 측에서 포지션이 변경되었음을 나타냅니다. 포지션 볼륨, 오픈 가격, 손실 중지 및 이익 획득 수준을 변경할 수 있습니다. 변	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	경사항에 대한 데이터는 OnTradeTransaction 처리기를 통해 MqlTradeTransaction 구조에 제출됩니다. 거래 실행으로 인한 포지션 변경(추가, 변경 또는 마감)으로 인해 TRADE_TRANSACTION_POSITION 트랜잭션이 발생하지 않습니다.	
TRADE_TRANSACTION_REQUEST	거래 요청	MqlTradeTransaction

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	이 서버에서 처리되었다는 알림 및 처리 결과를 받았습니다. MqlTradeTransaction 구조의 해당 트랜잭션에 대해서는 유형 필드(거래 트랜잭션 유형)만 분석해야 합니다. 추가 데이터를 얻으려면 OnTradeTransaction(요청 및 결과)의 두 번째 및 세 번째 매개	

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	변수를 분석해야 합니다.	
TUESDAY	화요일	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TYPE_BOOL	bool	MqlParam
TYPE_CHAR	char	MqlParam
TYPE_COLOR	색상	MqlParam
TYPE_DATETIME	datetime	MqlParam
TYPE_DOUBLE	double	MqlParam
TYPE_FLOAT	float	MqlParam
TYPE_INT	int	MqlParam
TYPE_LONG	long	MqlParam
TYPE_SHORT	short	MqlParam
TYPE_STRING	string	MqlParam
TYPE_UCHAR	uchar	MqlParam
TYPE_UINT	uint	MqlParam
TYPE_ULONG	ulong	MqlParam
TYPE_USHORT	ushort	MqlParam
UCHAR_MAX	uchar 유형으로 나타낼 수 있는 최대값	숫자 유형 상수
UINT_MAX	Maximal value, which can be	숫자 유형 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	represented by uint 우형으로 나타낼 수 있는 최대 값	
ULONG_MAX	ulong 유형으로 나타낼 수 있는 최대 값	숫자 유형 상수
UPPER_BAND	상한	지표선
UPPER_HISTOGRAM	상위 히스토그램	지표선
UPPER_LINE	윗선	지표선
USHORT_MAX	ushort 유형으로 나타낼 수 있는 최대 값	숫자 유형 상수
VOLUME_REAL	거래 볼륨	가격 상수
VOLUME_TICK	틱 볼륨	가격 상수
WEDNESDAY	수요일	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
WHOLE_ARRAY	배열이 끝날 때	기타 상수

© 2000-2025, MetaQuotes Ltd.

상수	설명	사용법
	가지 남아 있는 항목 수를 의미하며 즉, 전체 배열이 처리됩니다	
WRONG_VALUE	상수는 암시적으로 캐스트를 어떤 열거 유형이든 할 수 있습니다	기타 상수

© 2000-2025, MetaQuotes Ltd.